TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

David Hästbacka

Developing Modern Industrial Control Applications:
On Information Models, Methods and Processes for Distributed Engineering

Julkaisu 1143 • Publication 1143

Tampere 2013

Tampereen teknillinen yliopisto. Julkaisu 1143
Tampere University of Technology. Publication 1143

David Hästbacka

Developing Modern Industrial Control Applications:
On Information Models, Methods and Processes for Distributed Engineering

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Festia Building, Auditorium Pieni sali 1, at Tampere University of Technology, on the 9th of August 2013, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2013

ISBN 978-952-15-3098-2 (printed)
ISBN 978-952-15-3113-2 (PDF)
ISSN 1459-2045

Abstract

Control applications are used in automation and control of manufacturing and processing facilities to run the process. Development of industrial control applications is engineering with multidisciplinary characteristics and it is often performed as a part of a larger project, e.g. constructing a facility. It is closely interwoven with several related engineering disciplines between which information exchange is required. The control domain has several requirements and characteristics that need to be considered when developing new means to improve quality and efficiency of engineering.

In this thesis control application development is studied concerning model information content, methods for enhancing development, and improvement of engineering process management. Based on requirements for information content in control application models, improvements to the UML Automation Profile (UML AP) are presented. UML AP requirement modeling is developed to enable concerns of related engineering to be taken into account. Domain-specific UML AP constructs are also developed for modeling platform independent functions and execution platform features.

To improve development of control applications, model-driven methods are proposed and applied using UML AP elements. In the approach, a workflow from requirements to functional models, and finally to executable applications, is developed. The approach has been designed so that tools can be provided for automating capturing of requirements and assisting in model transformations. For development of control applications the approach promotes reusable, platform independent solutions while maintaining support for existing well-proven implementation platforms.

Methods are proposed for extending modeling with ontology descriptions based on Semantic Web technologies. Using these, a layer is constructed on top of the application model which is used to enhance interoperability and understandability of the concepts. The semantic descriptions enable automatic reasoning and inferences to be used in model analysis and provision of material supporting engineering. As a supplement to traditional modeling the descriptions provide semantics beyond those of a metamodel.

Given that the information content is standardized and the development methods are defined, the engineering processes can be improved concerning information exchange and process management. Organizing engineering tasks into services is proposed, and a service infrastructure is developed to facilitate utilization of these services and integration of information systems. Business process modeling is used to describe the engineering processes and to compose the services. By executing the business processes

with information systems the management of engineering activities is improved and automation of some of the tasks is enabled.

Engineering is not limited to the design phase of a facility. It is performed throughout the plant lifecycle in, for example, operation and maintenance (O&M). A service framework providing access to relevant data is also beneficial for O&M tasks. To meet the business demands a business process driven approach for development of O&M information systems is proposed. The approach is based on composition of services, and enables flexible reconfiguration of processes as well as integration of systems.

Keywords: control application, software model, model-driven engineering, ontology knowledge, engineering support, service framework, engineering process

Preface

The work presented in this thesis was carried out at the Department of Automation Science and Engineering at Tampere University of Technology. The research work was conducted during the years 2008-2012. During that time I had the opportunity to participate in several research projects that provided a framework for forming the objective and basis of this thesis. The funding of TUT President's Doctoral Programme is also gratefully acknowledged.

First of all, I want to express my gratitude to my supervisor Prof. Seppo Kuikka for his guidance, dedication, and support during the research. I am also very grateful for the interesting research topics and other responsibilities that have inspired me to broaden my horizon.

I want to thank the pre-examiners Prof. Kari O. Koskinen from Aalto School of Electrical Engineering and Dr. Alois Zoitl from fortiss GmbH An-Institut Technische Universität München for reviewing my thesis. I also wish to thank Prof. Valeriy Vyatkin from Luleå University of Technology for being my opponent at the public examination.

I wish to thank the professors, colleagues, and the personnel of the Department of Automation Science and Engineering for the motivating research and work environment. I am thankful to the Automation Software research group and the people that I have had the privilege to work with, both past and present. I want to thank especially Timo Vepsäläinen, Petri Kannisto, Jari Rauhamäki, and Outi Rask.

I would like to thank the people from Aalto School of Electrical Engineering and VTT Technical Research Centre of Finland that I have had the opportunity to collaborate with. I also want to thank the people from the industry that I have had the chance to work with and discuss matters of interest.

Many thanks also to my friends for the extracurricular and recreational activities.

I am grateful to my parents and my sister for supporting and encouraging me throughout my whole life.

Finally, I want to thank Anuliina for all the support and patience during this process.

Pirkkala, 17th May 2013

David Hästbacka

Contents

List of Included Publications

The thesis is based on the following publications referred to as [P1]-[P7].

[P1] Hästbacka, D., Mätäsniemi, T. (2009) Unifying Process Design with Automation and Control Application Development – An Approach Based on Information Integration and Model-driven Methods. Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing. Moscow, Russia, June 3-5, 2009, pp. 1227-1232. DOI: 10.3182/20090603-3-RU-2001.00204

[P2] Hästbacka, D., Vepsäläinen, T., Kuikka, S. (2011) Model-driven Development of Industrial Process Control Applications. Journal of Systems and Software, vol. 84, no. 7, pp. 1100-1113. DOI: 10.1016/j.jss.2011.01.063

[P3] Hästbacka, D., Kuikka, S. (2011) Bridging UML Profile Based Models and OWL Ontologies in Model-driven Development – Industrial Control Application. International Workshop on Future Trends of Model-Driven Development, Beijing, China, 8-11 June, 2011 Setubal, SciTePres, pp. 13-23. DOI: 10.5220/0003561900130023

[P4] Hästbacka, D., Kuikka, S. (2013) Semantics Enhanced Engineering and Model Reasoning for Control Application Development. Multimedia Tools and Applications, vol. 65, no. 1, pp. 47-62. DOI: 10.1007/s11042-012-1134-9

[P5] Hästbacka, D., Kuikka, S. (2009) A Service Oriented Engineering Approach To Enhance The Development of Automation and Control Systems. Proceedings of Information Systems Analysis and Specification, 11th International Conference on Enterprise Information Systems, Milan, Italy, May 6-10, 2009, pp. 219-224. DOI: 10.5220/0002007702190224

[P6] Hästbacka, D., Kuikka, S. (2012) Facilitating Services and Engineering Process Management in Distributed Engineering of Control Applications. Proceedings of the 10th International Conference on Industrial Informatics, Beijing, China, July 25-27, 2012, pp. 51-56. DOI: http://dx.doi.org/10.1109/INDIN.2012.6301198

[P7] Hästbacka, D., Kannisto, P., Kuikka, S. (2011) Business Process Modeling and SOA in Industrial O&M Application Development. Proceedings of the 13th International Conference on Enterprise Information Systems, Beijing, China, 8-11 June, 2011, vol. 3, Setubal, SciTePress, pp. 277-285. DOI: 10.5220/0003507202770285

List of Supplementary Publications

The following supplementary publications are related to the topic of the thesis but are either extended in the included publications or out of scope, and thus not included as a part of the thesis.

- Hästbacka, D., Kuikka, S. (2012) Semantics and Reasoning for Control Application Engineering Models, Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science, vol. 7267, pp. 647-655.

- Vepsäläinen, T., Hästbacka, D. & Kuikka, S. (2009) A model-driven tool environment for automation and control application development - transformation assisted, extendable approach. Proceedings of 11th Symposium on Programming Languages and Software Tools and 7th Nordic Workshop on Model Driven Software Engineering, 26-28.8.2009, Tampere, Finland. pp. 315-329.

- Vepsäläinen, T., Hästbacka, D., Kuikka, S. (2010) Simulation assisted model-based control development - unifying UML AP and Modelica ML. MESM' 2010, 11th Middle Eastern Simulation Multiconference Gameon-Arabia'2010, December 1-3, 2010, Alexandria, Egypt pp. 43-50.

- Hästbacka, D., Laitinen, O., Tommila, T., Kuikka, S (2007) Implementing a Work Support and Training Tool for Control Engineers, Proceedings of the 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Dortmund, Germany, September 6-8, 2007, pp. 512-517.

List of Abbreviations

API	Application Programming Interface
BPMN	Business Process Model and Notation Business Process Modeling Notation
CAD	Computer-aided Design
DCS	Distributed control system
DL	Description Logics
DSL	Domain-Specific Language
DSM	Domain-Specific Modeling
DPWS	Devices Profile for Web Services
EMF	Eclipse Modeling Framework
ERP	Enterprise Resource Planning
FBD	Function Block Diagram
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
MES	Manufacturing Execution System
MOF	Meta Object Facility
MDA	Model-Driven Architecture
MDE	Model-Driven Engineering
MDD	Model-Driven Development
MDSD	Model-Driven Software Development
OEM	Original Equipment Manufacturer
OWA	Open World Assumption
OWL	Web Ontology Language
P&ID	Process and Instrumentation Diagram
PID	Proportional-Integral-Derivative (control algorithm)
PLC	Programmable Logic Controller
RDF	Resource Description Framework
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SysML	Systems Modeling Language
UML	Unified Modeling Language
UML AP	UML Automation Profile
URI	Universal Resource Identifier
URL	Uniform Resource Locator
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformations

List of Figures

1 Introduction

This chapter introduces the topics of the thesis and provides the background and motivation for this thesis. The research questions discussed are outlined, the scope of the thesis is defined, and the research methodology is introduced. After that, the contributions are presented, and finally, the organization of the thesis is outlined.

1.1 Background and Motivation

The responsibilities of information systems used in manufacturing range from enterprise level management and plant wide scheduling of production activities to controlling single actuators or reading sensors measuring properties of interest. Industrial control systems, often referred to as distributed control systems (DCS), are responsible for controlling and monitoring the functioning of production processes in various industries. Control systems have become important for the performance and reliability of manufacturing as intelligent control applications have gradually replaced the need for human operators.

Typical to modern control system development is implementing applications using software based solutions, and executing them in a distributed manner composed of networked components and subsystems. The control applications are software programs responsible for gathering and analysing measurements, processing data, calculating actions and values, and controlling equipment and devices in order to steer the process towards the desired outcome.

The lifecycle of an industrial control application typically starts with identified business needs and constraints. These initial requirements go through a number of design and implementation phases, including testing and verification, to finally reach production as a running application. Even though a significant part of the engineering is executed in the design phase, engineering is also present in operation and maintenance of the running facility, and eventually also in the modernization or disposal of the plant. The design of an industrial plant is a huge assignment that interweaves several engineering disciplines with the topic of this thesis, control application engineering, for example process design, and electrical and instrumentation design.

The design of plants, as well as the engineering of control applications, is networked between teams of engineers of various disciplines that can also be distributed among different companies. Decentralized development and the strive for cost-effectiveness

have a negative impact on flexibility and collaboration in engineering networks, and impose information management challenges to engineering processes and to different information systems and tools used in engineering.

The engineering effort and complexity of control applications has increased as the responsibilities of control applications have grown. In addition, this has increased the amount of engineering data being produced, management of which needs new methods in order to better support application engineering. The tools and technologies used in control application development rely on utilization of information technology. To improve efficiency and support engineering new advanced methods can be developed that integrate engineering concepts, tools and engineering processes. For new modeling concepts and methods, however, enabling infrastructure architecture and accompanying systematic procedures are required as well. An important requirement for developing new methods is to reduce the effort required in managing knowledge and application model content due to the increasing number of features and complexity in applications.

Providing design data in a machine interpretable or semi-formal format promotes system integration, and increases future prospects of utilizing the information in the engineering lifecycle. This also enables developing new features and services in tools and information systems that support engineering more efficiently, e.g. by integrating design data and models with external information sources, analysis services, and by coupling support to management of engineering processes (Figure 1). In addition, the engineering information can be used to further extract data for use in engineering tasks during maintenance operations of the running facility, e.g. routine maintenance work, and modernization and restructuring of the facility. This is of importance due to the fact that production facilities have shifted their focus on core production activities, and many of the supporting tasks are outsourced. As a result, information exchange challenges have emerged in consumption and provision of expertise and services of external service providers.

The motivation for this thesis is to study how engineering of control applications could be enhanced in order to improve engineering efficiency, increase quality of applications, make management of engineering processes more effective, and reduce total costs of development.

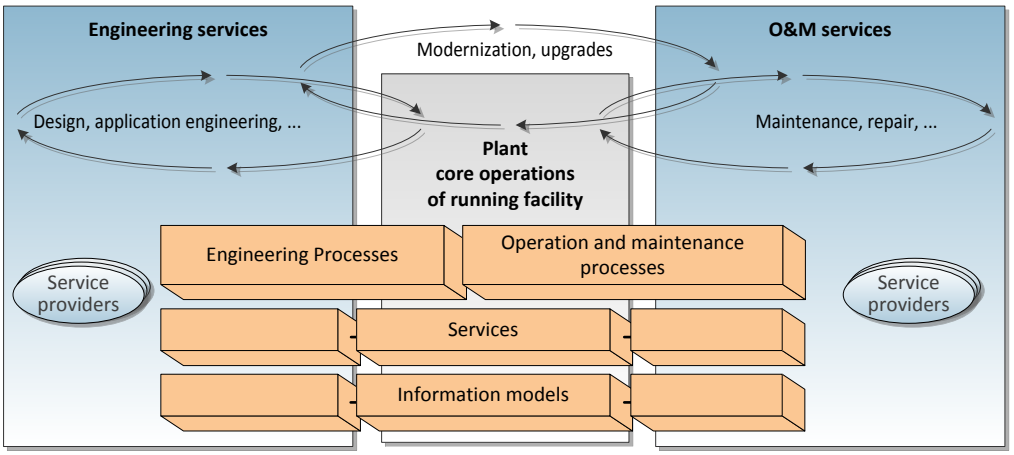

Figure 1 An overview of plant operations from the point of view of services and plant engineering information.

1.2 Research Questions

The thesis explores what kind of application models, development methods and processes can be used to enhance distributed engineering of control applications for modern control systems that are implemented as complex software applications. For this it is studied how information should be described in application models adhering to standards and consistency. Additionally it is studied what kind of development methods and process management could be used in the engineering lifecycle to support and improve development in distributed environments of multiple service providers and collaborating participants.

The research topic is divided into three themed groups of research questions referred to as RQ1-3.

1. What kind of engineering information is required in development for the industrial control application domain? What requirements are there on concepts and constructs used in modeling from a distributed development perspective? How can interoperability of models and concepts be improved to support development during the engineering lifecycle?

2. What are the characteristics of development methods that enhance distributed engineering in the domain? Can development be systematized with model-driven methods to improve transfer of multi-disciplinary information and what

are its implications on application development? How can engineering concepts of the systematized process be used to support development?

3. What are the requirements for automating engineering processes? How can engineering processes be managed and automated using services and business process modeling methods? Are the concepts and development methods proposed above applicable in improving management of distributed engineering activities and automating some of the tasks?

The author argues that with the proposed models, adhering to standards and logical consistency, the control application design can be communicated between designers of various disciplines in a distributed engineering network. In addition, the consistent model information content allows transformations and output to programmatically interpretable models, e.g. for automated management and improved engineering efficiency. Based on this, it is further argued that design as well as later operation and maintenance can be supported with services utilizing this information. On the basis of business process modeling of engineering activities, and the combination of executable processes with services, it is suggested that management of workflows can be improved and automation of straightforward tasks can be achieved.

1.3 Scope of the Thesis

This thesis discusses development of control systems and engineering of applications used to control industrial processes and manufacturing. The main focus is on how engineering can be enhanced in a distributed network of multiple participants with models and a service-based approach, and how the use of these models can be extended during the whole engineering lifecycle. The engineering lifecycle refers to the engineering work performed during development of the system and during operation and maintenance of the running facility. The use of models for applications in execution is not considered in this thesis.

Models of control applications are studied from the point of view of developing the software artefacts implementing the needed functionality. Plant information models as such are only used as input to position the control application model and its parts to the logical plant hierarchy. The development of plant information models and research related to plant modeling is considered to be out of the scope of this thesis.

In the thesis, services are considered as organizational units of offerings of engineering work, whether they are implemented as human labour or with information systems, and

as consumers and providers of information content in the application models. The use of services is studied from the viewpoint of automating design tasks, enabling provision of specific expertise, and facilitating the use of this expertise in development.

Engineering processes are only considered to the extent required for improving management and automating certain engineering tasks based on the application models and the service based operation approach proposed. The focus is on the application model information content in distributed engineering, modeling methods of the engineering processes, and the technologies and methods used to execute processes and improve their management.

The thesis primarily considers engineering during design, but the role of design time models and plant information in operation and maintenance of running systems is also discussed. The design time data is an important asset when operating the plant and access to this data is vital in many activities. It can be argued that standards-compliant models and practices during design are a prerequisite for efficient later utilization. Also, in the future it may become even more important when considering application integration and the increase in more intelligent software applications that can utilize this information programmatically.

1.4 Methodology

The methodology followed in this thesis is based on constructive research of design science. The construction begins with design thinking and projections into the future solution with theories and artefacts. Conceptual and knowledge gaps are filled with purposefully tailored building blocks to support the constructions, e.g. models, diagrams, algorithms, and software development methods. When a construction differs from previously existing ones, whether it is a theoretical or practical one, it constitutes a new reality against which pre-existing ones can be examined and understood [21].

Owen [82] presents a general model for understanding the design research process where artefacts and action are essential in the process of generating and accumulating knowledge. The process is presented as a cycle in which knowledge is used to create artefacts, the artefacts are evaluated to build knowledge, and channels between the two define the conventions and rules for the knowledge generation.

The outputs of design research can be defined as constructs, models, methods, and instantiations according to March and Smith [56]. Purao [89] has defined similar outputs but constitutes in addition an important fifth output: better theories.

The research methodology used in this thesis has also characteristics of contextual qualitative research as pragmatic work has been performed in various research projects related to and in close collaboration with the industry.

The main research steps are as follow:

1. Definition of the domain and related disciplines. Specification of requirements of the target domain (e.g. reuse and existing technological background) to be taken into account. Definition of information flow and dependencies between engineering disciplines.

2. Requirements for application constructs are conceptualized for control software applications. The application constructs are specified with requirements of the domain and dependent disciplines in mind. A model-driven approach, based on the developed constructs, is developed for application and system engineering.

3. The use of ontologies as a supplement to modeling constructs is investigated to enhance the information associable with application models and to improve model semantics. Methods are developed for concurrently using ontology descriptions with application models, and to integrate these descriptions with other engineering knowledge.

4. Service based methods for incorporating better interoperability to a distributed engineering environment are conceptualized, and opportunities of improving engineering processes are studied.

5. The systematized development approach and proposed engineering information content is utilized and a business process modeling method is applied to model and manage engineering processes. The management of engineering processes and automation of tasks is studied using engineering process descriptions and a service based approach of processing engineering data.

The evaluation of the results is based on assessment in each step with questions regarding fulfilment of requirements, comparison to existing solutions, and evaluation of improvements and the added value the new constitution brings.

1.5 Contributions of the Thesis

The main contributions of the thesis in respect to the research questions are following.

RQ1 Information Models:

- Requirements on data in a shared plant information model for distributed control application design, e.g. in order for information related to application models to be shared in the distributed environment.
- Specification of domain-specific aspects of industrial control application development for model-driven engineering, and contributions to the UML Automation Profile modeling constructs for modeling control application requirements and functionality.
- Alignment of MOF based metamodeling with OWL ontology modeling to further the engineering environment for the Semantic Web. Development of two different methods for producing corresponding OWL ontology descriptions of application models being designed.
- A method for using OWL ontologies as a supplement to application models to describe additional features of the model, and to enable integration of other engineering knowledge. The method includes a categorization of design information used in engineering scenarios to promote semantic integration and management of knowledge.

RQ2 Development Methods:

- Specification of requirements and methods for implementing information flow in a model-driven approach for systematized control application development.
- Contributions to the specification and implementation of plug-in components for the developed prototype tool environment for model-driven development of industrial control applications.
- An approach for applying engineering knowledge to categorization and analysis of control application models using Semantic Web technologies, and an architecture for implementing concurrent semantic processing in the integrated development environment.

RQ3 Services and Processes:

- An outline of using a service-based approach to organize engineering activities, and definition of implementation requirements on engineering information for using services during the engineering lifecycle.

- A specification and prototype implementation of a service broker architecture to facilitate the use of services in distributed engineering environments.
- A proposal of using a business process modeling method to model and manage engineering processes, and evaluation of its applicability for industrial control application engineering.
- Proof of concept implementation of reusable executable workflows between services to automate tasks in control application engineering processes.
- Requirements on design information and services, and evaluation of using business process modeling based methods for developing compositional O&M information systems.

1.6 Organization of the Thesis

The research questions of the thesis are addressed in separate chapters as presented in Table 1. The table indicates the included publications concerned and their order of importance in relation to each topic of research questions.

Table 1 he included publications, the research questions and their presentation in this thesis.

Research Questions	Publications	Thesis chapter
1 Information Models	P1, P2, P3	3
2 Development Methods	P2, P1, P4	4
3 Services and Processes	P5, P6, P7	5

Chapter 2 gives an introduction to the technologies applied in the thesis and in the publications. The technologies presented are general purpose information technology methods and concepts that have been used and applied to the needs of control application development. The models and concepts proposed for representing information content in industrial control application models are discussed in chapter 3. The development methods aiming to improve information exchange and engineering efficiency are presented in chapter 4. Means to more efficiently manage the distributed engineering activities and automate certain tasks using services and process management are presented in chapter 5. In chapter 6 a summary of the included publications is given. The contribution of the author in each of the included publications is also explained. Chapter 7 concludes the thesis with a re-examination of the research questions and an outlook on future research.

2 Technological Background

The main focus of the thesis is on engineering of control applications that are being executed on DCS platforms or in programmable logic controllers (PLC). Control systems of this kind are, for example, used for real-time control of processes in manufacturing, chemical industry or power plants. Control applications developed for DCS platforms include monitoring and control of process values as well as advanced regulatory control functions. Modern DCS platforms often provide additional features related to supporting activities such as collecting historic data, quality control, condition monitoring features, and inventory control of e.g. consumed resources.

DCSs are typically set up for individual processes or machines and integrate the devices and equipment needed in operating the process. In the hierarchy of plant information systems DCSs are set below the systems referred to as manufacturing execution system (MES) and enterprise resource planning (ERP). While DCSs are primarily used for real-time regulatory control, the MES and ERP system functionality is inclined towards production resource management, production scheduling and business transaction level operations.

Following is a brief introduction to the technologies and methods used in the thesis for developing new means to improve control application development.

2.1 Software Modeling and Model-driven Development

2.1.1 Metamodels and Meta-Object Facility

Metamodels are models that define the rules for modeling and describe the possible structure of models. A metamodel can be used for defining a domain-specific modeling language, to validate models, and to perform transformations between models, for example. Metamodels and models have a class-instance relation, i.e. models are instances of metamodels. Metamodels can in turn have metametamodels that describe the rules for defining metamodels. [107]

Meta-Object Facility (MOF) is an example of a standardized metamodeling architecture that defines four metalevels. The M0 level represents real world objects that M1 level models describe, i.e. M0 level objects are instances of M1 models. Similarly the M2 level defines the constructs available for use in M1 models. Applied to UML the M2 level translates to constructs such as the Class, for example. Continuing the abstractions the UML Class is an instance of the meta meta element MOF Classifier defined in M3.

To avoid continuing the abstractions from M3 the MOF is defined using MOF constructs, i.e. it defines itself. [78][107]

2.1.2 UML

Unified Modeling Language (UML) is a visual language that has both a syntax and semantics. A major revision to the modeling method was introduced with UML 2.0 when the infrastructure metamodel was specified [75]. Common uses for UML are designing software, documenting systems, communicating software or business processes, and capturing details about a system for requirements or analysis. UML is most commonly used for designing software but it can be applied to many other areas as well. For modeling UML provides different diagram types for both generic uses and for specific purposes, e.g. for modeling structure, behaviour and interaction. UML also defines a profile mechanism for extending the metaclasses for different purposes, i.e. a particular domain or a family of applications. [87][77]

2.1.3 SysML

Systems Modeling Language (SysML) [72] is a general purpose modeling language similar to UML. The modeling language is targeted for systems engineering applications and extends some of the UML 2 concepts using the profile mechanism. SysML supports specification, analysis, design, and verification and validation of systems including, for example, hardware, software, information, processes, personnel, and facilities.

2.1.4 Domain-specific Modeling

Domain-specific modeling (DSM) is the idea of creating models for a domain in a domain-specific language (DSL) that suits the needs of the particular domain [107]. Generic concepts of traditional modeling languages easily make the models large, complex, and difficult to understand for domain professionals. One of the major benefits of developing and using DSM is to raise the level of abstraction in development using concepts of the domain. A DSL allows extracting the most important aspects to be used in modeling, hence allowing design to focus on key issues and mission critical functionality.

2.1.5 Model-driven Development

Model-driven engineering (MDE) and model-driven development (MDD) promote models as primary and most important engineering artefacts during development. In the development process models are refined through a number of iterations where the models are elaborated and details are added from different viewpoints. Model transformations, either manual or automatic, are used between the iterations to create the basis for the next model iteration. Finally the resulting model, or group of models, is the expected outcome that becomes the executable application, e.g. as generated and compiled program code. For example, Model Driven Architecture (MDA, [79]) is a well-known software development approach based on the use of models.

2.2 Semantic Web

2.2.1 Concept

The idea of the Semantic Web is to provide an extension to the Web in which information is given a well-defined meaning allowing computers and people to work in cooperation [9]. In this vision knowledge is shared in an open environment, and machine interpretable semantics allow automated agents and software applications to access resources in an intelligent fashion. An important aspect of semantic descriptions is ontologies, i.e. collections of information that are used to specify taxonomies for classes of objects as well as their relationships and properties.

2.2.2 Technologies

The Semantic Web makes use of a number of technologies for accessing and describing resources. Regular Web content is usually presented in Hypertext Markup Language (HTML) or in eXtensible Markup Language (XML). The most basic technology to add meaning to information is to use the Resource Description Framework (RDF). RDF encodes knowledge in triples similar to the subject-predicate-object structure in sentences. All objects on the Semantic Web are identified by a Universal Resource Identifier (URI). Also relationships between objects as well as the object properties are defined using concepts that are denoted by an URI. These concepts can then be shared and linked globally. In triples URIs are used to ensure that concepts are tied to a unique definition available on the Semantic Web [9].

The Web Ontology Language (OWL) is an ontology language for the Semantic Web. OWL provides concepts such as classes, properties, individuals, and data values to

describe objects and resources in ontology documents. OWL 2 refers to the updated specification from 2009 that introduces a number of new features compared to OWL 1. These include OWL profiles, keys, property chains, richer datatypes and data ranges, qualified cardinality restrictions, and new kinds of property relations. [122]

Other technologies associated with the Semantic Web are, for example, RDF Schema (RDFS) to describe RDF vocabularies, and the SPARQL Protocol and RDF Query Language for querying data stored in RDF. XML is often used for the ontology syntax, e.g. RDF/XML and OWL 2 XML, but other exchange syntaxes exist as well.

2.3 Service-oriented Architecture

2.3.1 Overview

Service-oriented architecture (SOA) is an architectural paradigm for information systems that aims to solve integration and interoperability issues, e.g. in enterprise systems. SOA strives for encapsulation of functionality into reusable service components with well-defined interfaces. Services are loosely coupled in SOA, and they can be discovered, bound and invoked in an adjustable and standards-compliant manner.

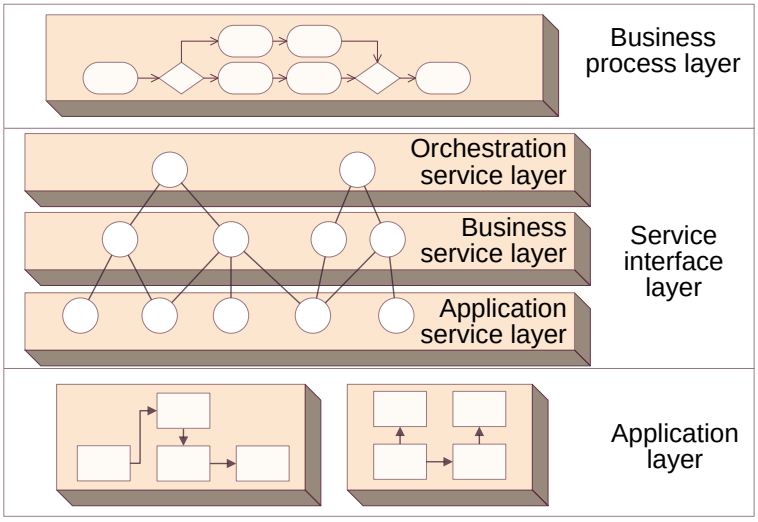

Figure 2 Alignment of business processes, service interface layers and applications. [25]

Figure 2 illustrates SOA composition of services in relation to business processes [25]. The top level business processes represents actual physical tasks and interactions performed in daily operations. The orchestration service layer stands for the implemented and executable process definitions corresponding to and supporting the real world processes. The layer can be seen as the compositional layer integrating

services but also other processes as workflows. The business services represent logic of business models, and the application services integrations to information systems, tools, and data sources. In addition, utility and adapter services are often required to implement the compositions.

2.3.2 Web Technologies

A service-oriented architecture is often related to Web services in which content is accessed and manipulated using standard Web technologies. Web service interfaces can be described using the Web Services Description Language (WSDL) [124] in order to define a machine processable description of how an interaction with the service can be established. A service consumer invokes the service with a request that the service provider processes, and finally returns a response to. This communication is executed using the XML based Simple Object Access Protocol (SOAP) [123] carrying the actual XML based message content, and transmitted using the Hypertext Transfer Protocol (HTTP). Another way to implement Web services is the Representational State Transfer (REST) [27] based approach. It is a client-server model in which requests and responses include representations of resources.

The two approaches differ in several ways and both can be justified for many purposes. SOAP Web services can be seen method oriented compared to the REST approach that focuses on resources. REST endorses a general uniform interface and HTTP is often used as the protocol for accessing and manipulating resources. SOAP based Web services easily become more intricate than REST style services that by design are more lightweight. An example of this is the stack of WS-*[1] technologies that, nevertheless, may be useful in cases where, for instance, more detailed messaging specification, service metadata exchange, or security features are required.

A service-oriented architecture is often implemented using a combination of different technologies. Other technologies that are commonly used are, for example, WCF, DCOM, DDS, and CORBA.

[1] WS-* refers to the associated SOAP Web services technology specifications such as WS-Addressing, WS-Discovery, WS-Eventing, WS-Policy, and WS-Security.

2.4 Business Processes Modeling

2.4.1 BPMN

Originally BPMN stood for Business Process Modeling Notation but was renamed to Business Process Model and Notation for version 2.0 [68]. BPMN is a notation for business process modeling, and it is aimed to increase understanding of business transactions and collaboration inside and between organizations. The modeling is based on pools and lanes that group process participants. Inside these tasks are defined that the participants execute according to a workflow that connects the tasks using gateways and event transitions. In addition, message flows can be used to describe information exchange between tasks and participants, and objects can be linked to the tasks e.g. to represent other artefacts of interest. BPMN versions prior to 2.0 were considered mainly as a graphical notation before the 2.0 specification defined the syntactical structure, and formalized the execution semantics for elements enabling running the processes in compatible business process execution engines [18].

2.4.2 Business Process Execution

To execute business processes modelled using BPMN prior to version 2.0, or some other solely graphical notation, they had to be implemented in information systems using other means. One way to implement business processes is to use traditional programming languages to build requests between systems and components directly into the clients and services according to the process descriptions.

However, if the information systems implement a service architecture, the services can be composed into orchestrations e.g. using the Web Services Business Process Execution Language (WS-BPEL [66]). Although developed for different purposes, WS-BPEL has resemblance with BPMN and to some extent BPMN diagrams can be automatically transformed to WS-BPEL orchestrations [60][P7]. Since the introduction of business process execution semantics in BPMN 2.0 the execution capabilities have been enhanced and support for round-trip engineering of executable processes has improved.

3 Control Application Information Modeling

Control applications represent the logic that automatically operates a process in a facility. For typical DCS or PLC platforms the applications include monitoring and control of process values as well as complex regulatory control functions. Control applications usually also enable human operators to monitor and interact with the applications, i.e. changing settings and parameters. The engineering of these applications is related to other engineering disciplines during the engineering of a facility, e.g. process design or instrumentation. Control applications integrate equipment and devices used in measurements and for manipulating the physical process, and the application control loops are related to the process engineering functions. [45]

In the field of automation of manufacturing plants AutomationML [23] strives to provide an integrated data exchange format compatible with the different phases of engineering. It utilizes existing standards such as CAEX, COLLADA, and PLCopen XML. The approach is detailed but limited to tools and implementation platforms supporting the utilized technologies. The IEC 61131-3 [44] standard specifies the structure of programming languages used for developing control applications for PLC platforms. The specification, however, mainly deals with generic programming language constructs and the configuration of programs. IEC 61499 [43] is a similar open standard focusing on the function block composition of distributed control functionality with several improvements compared to the IEC 61131-3 model. Although it has been actively studied and used as a reference model by the academia it has not yet been adopted by the industry to the same extent as IEC 61131-3, for example.

The use of Semantic Web technologies for data interoperability and industrial systems integration has been considered. For process engineering data an ISO 15926 [46] based upper ontology has been proposed [6], i.e. a domain ontology to define common binding concepts. To integrate chemical process engineering information a formal OntoCAPE ontology has been presented [128]. Another example is the OntoSTEP [4] approach to use OWL ontologies to enrich product models so that a semantic knowledge base can be consolidated, i.e. for CAD geometry information represented in STEP[2] combined with other knowledge. For DSLs an approach has been proposed in which ontologies are used as an integrated language to reason on the DSL [125, 126].

[2] STandard for Exchange of Product model data (STEP) (ISO 10303) [88]

Many of the approaches, for example STEP and ISO 15926, have turned out to be too unspecific or complex in order to meet the requirements for standardized data exchange across companies [128]. Linked Data [11] has been proposed as another solution to connect information and vocabularies but as a generic method there is no consistent information metamodel or other means to assure, for example, integrity of this data. [30]

In this chapter requirements on models and engineering information related to control application development are discussed. Based on these a modeling method is presented that provides modeling constructs for control application development. Finally, the use of ontologies as a supplement to these models is introduced.

3.1 Requirements on Model Information Content

3.1.1 From Preceding Design to Control Application Design

Operation of continuous industrial processes is an important application area for advanced control systems. In this thesis, process design is used as an example of how requirements are gathered and elaborated for control applications, and what kind of information content the models need to represent. The current practices in process and automation software design have been studied in [P1] to understand what kind of information is transferred from process engineering to control application design.

During the preliminary process design needs and requirements for material processing are collected, and process block diagrams and charts are created. The diagrams describe for example chemical and physical transformations with major inputs and outputs, and the presented process variables are candidates for monitoring and controlling the process. These diagrams contain conditions that have to be satisfied during processing and also present what has to be done and in which order. Conditions can represent the processing limits in different production phases, alarm and interlocking thresholds, or transition triggers in sequences.

In basic process design the processing phases are allocated to units that have capabilities and processing operations required in the production. In this phase the first estimates of unit capacities are specified which are then used to guide equipment selection. One of the most significant results from basic process design is the schematic presentation in piping and instrumentation diagrams (P&ID). These diagrams present processing units with auxiliary systems, and requests to electrical, instrumentation and control design. The functional topology of the plant, presented in the P&IDs, then guides the different

engineering disciplines, e.g. measurement and control points for control engineering and level switches and sensory requirements for instrumentation.

Finally, in the detailed process design phase the requests are elaborated with additional constraints and conditions of process materials, processing conditions, accuracy requirements, mechanical connectivity, safety, and equipment options, if requested by the end customer. These constraints and conditions are typically collected to various function lists, equipment datasheets, and alarm and interlocking lists. The information exchange between process designers and instrument, control, and electrical engineers is accomplished with these kinds of lists and tables. From the control application development point of view the most important ones are the input-output (IO), function loop, and the instrumentation lists.

The design of automation and control functionality typically starts with a preliminary design phase in which a functional specification is created. The specification is a collection of loop or functional descriptions, but may also contain control strategy descriptions. In addition to design, the material can also be used by operators for studying or as guidelines in abnormal situations. For control application development the specification states requirements on functionality and criteria for acceptable operation.

In the basic design phase of automation and control engineering the design is of an implementation independent nature but platform specific details and implementation issues can appear and steer the development. In this phase control system functions are detailed, and control structures, sequences and control logics are defined. In the following detailed design phase the chosen control application platform affects development significantly when the application architecture is detailed and type circuits and control loops of the chosen platform are used to implement the functionality. At this stage of development also the configuration of instrumentation equipment as well as other connected systems is required.

3.1.2 Requirements on Data and Plant Information Models

The plant information model, that the control application model is also a part of, constitutes the logical information model of a production or manufacturing facility. A plant model can be defined in many ways depending on what is considered belonging to the model. At its broadest definition it is a complete virtual representation of the facility with all of its components, features, and properties. More generally it can be regarded as an information model describing the structure, the manufacturing or production process,

the equipment and devices, the control functionality, and processes and activities related to its operation during the plant lifecycle [6][14][51][88].

The plant model is typically organized as hierarchies dividing the plant into areas, segments, or units, for example [101][99]. During the entire design process it serves as a structured model to which design artefacts are logically linked as well as a means to organize the engineering work performed [7][98].

From a control application development point of view the following general requirements should be considered for plant information models in order to improve information exchange in distributed environments:

- The plant information model shall provide a coherent view of the logical plant hierarchy. It is important that different users of the plant model use the same ordering to organize and communicate design. The plant model provides slots to which engineering data and design artefacts are positioned.
- The plant information model shall support the use of various modeling methods and data formats in a neutral manner. Because of varying needs it is often required to use different tools and methods in engineering. Therefore the plant information model should not restrict the use of methods to allow choosing the most suitable methods for each task.
- The plant information model shall support different views to the data depending on the discipline and the role of the participating engineering company. The plant information model may provide access control features to permit or limit access to the data, e.g. to ease handling large amounts of data or to support work of competing companies in joint projects.
- The plant information model shall allow unified access to the data and accessing the data should not be method or tool specific. The information content needs to be structured and programmatically accessible for various tools and engineering system integrations.
- The plant information model needs to be up to date in order to provide accurate information necessary in engineering and control application development.

Ideally, a plant information model serves as a consolidating platform to integrate and distribute engineering related information throughout the lifecycle of the plant. It also supports working in project consortiums consisting of multiple distributed engineering teams encompassing different disciplines. For control application engineering the plant

information model provides the necessary information used in application development, e.g. data and diagrams from related engineering disciplines, measurement and control points, as well as instrumentation and equipment.

Many of the general requirements presented impose additional requirements on development and engineering practices throughout the engineering lifecycle. Data management processes, e.g. how to handle changes that affect decisions already made, are emerging issues that will be discussed in the following chapters in relation to control application development.

3.1.3 Domain-specific Aspects of Industrial Control Application Models

The characteristics of the automation and control domain affect what kind of information the control application models need to contain. These domain-specific characteristics originate from related engineering disciplines, and the nature of developing control applications and its history.

Information Originating from Related Disciplines

In engineering where multiple disciplines collaborate and depend on each other it is important to be able to connect pieces of information [P1]. For this it is required of the control application models to be able to contain data that links model functionality and features to common locations of the engineering. In its simplest form this can be achieved by using an agreed naming convention that includes the logical location in respect to the plant model. However, it is not uncommon to use multiple naming conventions, e.g. different during engineering and end-user operation, and support of this might be required in many cases.

Interpreting engineering information from preceding engineering and dependant disciplines typically requires extensive expertise [P1]. The diagrams and detailed lists of process engineering, for instance, require knowledge to understand and utilize the often implicit information during control application development [P2]. As control systems are used in many fields the target application area also has an impact on development. This requires knowledge of e.g. the nature of the process and the processing or manufacturing methods used. All these include information that is of relevance when developing control applications. Therefore the control application models should support including this information to application development in order to more explicitly specify requirements for development. Properly specified requirements also

reduce the amount of implicit information in control application development and improve traceability of design when the information chain is explicit.

Requirements originating from process design lists include parameters that represent conditions for operation [P1]. These kinds of requirements are typically of a restrictive nature limiting possible solutions as well as the selection of equipment and instrumentation that can be used. In general, requirements are of a functional or non-functional kind, and include textual descriptions. When elaborating control application requirements they can also be supplemental in the sense that they provide essential parameters or restrictions that need to be taken into consideration in control application development. For this it is beneficial for the control application models to support the use of representations of acknowledged standards or otherwise agreed practices. This is also a requirement because of the fact that there is a variety of tools and methods used that consume and produce information that needs to be consolidated.

Control Application Engineering and Existing Platforms

Current DCS platforms offer sophisticated features for application developers that enable focusing on the functionality of applications through function block networks [P2]. Typically the platform takes care of running the applications whether they are distributed on PLCs or executed inside a process station, and executes functions within limits of specified real-time constraints. The control system vendors also have a history of targeting their systems for specific application areas, and as a result balanced trade-offs between system properties such as performance, reliability, distributability, and application management. It can be argued that many of the current DCS platforms, the result of years and decades of development, offer approved and well tested features and capabilities that are worth supporting in future approaches.

Control applications have an additional interesting characteristic: collections of type circuits. Type circuits are parameterisable function-block-like types used in DCSs that represent reusable implementations of specialized functionality on the current platform. These type circuits are often used to control certain kinds of sensors or actuators, and to perform control based on signals provided by and linked from other type circuits, for example. The reusable type circuits, although typically restricted to a specific platform, often represent years of expertise captured in a well-tested solution.

New emerging control application design paradigms have not been able to replace traditional methods of developing control systems using DCS platforms nor have they gained foothold as one might have expected. This is probably due to many reasons but

one important is prudence and the slow adoption of new methods due to the critical nature of the applications. New development approaches should take these characteristics into consideration and, without reducing quality or unnecessarily complicating design, also utilize these as assets.

Engineering of automation and control functionality draws platform dependent characteristics early on in the development [P1]. On one hand, this is due to features and methods provided in tools of control system platforms and, on the other hand, the lack of acknowledged methods that support platform independent development. There are, however, many elements of platform independent design that should be emphasized in new development approaches in order to promote reuse of solutions and other utilization of application models.

Typical to industrial control applications is that besides being complex they are often large in number of components and application blocks [49]. Control applications that monitor and control typical industrial facilities can easily contain thousands of parameterisable function blocks. For development approaches and associated modeling methods it is therefore important to support vast amounts of engineering data.

On Model Utilization

Basically, control application models include the information of how the control system is constructed and how the system operates as executable programs. In principle, this information could be directly used for many purposes during the application lifecycle, e.g. for integration of new information and control systems or in dynamic compositions [110] of functionality in the next generation of intelligent manufacturing systems [113].

The most significant limitation of using control application models in the later lifecycle is modeling of the applications using platform specific constructs. In addition to differing methods for expressing functionality also the granularity used in the concepts varies, e.g. what is the level of features and capabilities of similar appearing concepts in different systems.

If a common method were available it should be defined on such a level that it is unambiguous and understandable to all parties, e.g. by enabling alignment to standardized definitions [P1]. Ideally, for the application models to be usable they also need to be accessible without restrictions on tools or development environments, and allow sharing not only in the own organization but also to other partners if required.

The control application models, as well as the plant model in general, need to be integrated to the business processes of the running facility in order to maintain valid and

up to date information throughout the lifecycle. It can be argued that the plant model and the control application models are assets of comparable importance to the physical system and its components. This requires well defined procedures of how the model is maintained, i.e. responsibilities during engineering, but also calls for processes to data and information management during operation.

3.2 UML Automation Profile

The UML Automation Profile (UML AP) was initially developed and presented by [94]. Since then the profile has been further developed as presented in [P2] with the previously discussed characteristics and requirements in mind. The profile serves as a prototyping environment and method using which new approaches for control software development can be tested and evaluated. In this thesis, the improvements to the profile are introduced and it is used as the basis for modeling control applications. This thesis does not include an extensive presentation of UML AP and the profile is discussed mainly in respect to the topic of this thesis. The use of the profile constructs in application development is also discussed in chapter 4 Engineering Methods for Industrial Control Applications.

UML AP has been developed to bring control domain specific elements to industrial automation and control software development. Profiles are the means of UML [75] to extend the modeling with domain-specific concepts and semantics, and the modeling elements of profiles typically consist of stereotypes, tagged values and constraints. The UML AP stereotypes are partially extended from suitable elements of the UML Profile for Schedulability, Performance and Time [76], SysML [72], and UML Profile for Quality of Service and Fault Tolerance [74]. The modeling elements have been constructed by examining and extracting development practices and concepts used in the domain. This has been done, on one hand, to allow familiar domain concepts to be used in developing solutions, and, on the other hand, to ensure compatibility to existing practices of the industry. The UML and SysML background is not apparent due to many custom elements and diagrams with domain resemblance. This is a benefit from the modeling point of view as designers can work with familiar concepts of the domain instead of generic software constructs.

The profile is organized into subprofiles that each addresses a specific part and perspective to control software design. The Requirements subprofile is for modeling requirements of different stakeholders, and the Automation Concepts subprofile for modeling domain-specific control application functionality. The Distribution and

Concurrency subprofile and the Devices and Resources subprofile provide concepts for modeling system level distribution of software and concepts for modeling devices of the execution platform, respectively. The subprofiles can be used also separately although the profiles support one another and the modeling concepts are designed orthogonal [94].

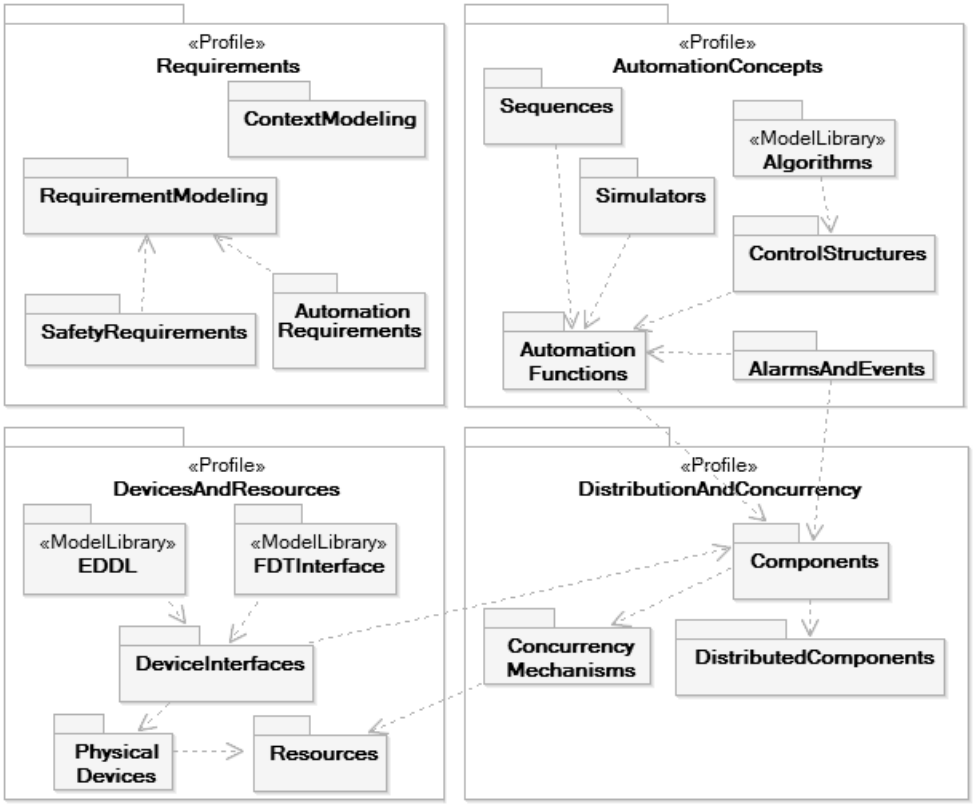

Figure 3 The UML Automation Profile consists of four separately utilizable subprofiles.

3.2.1 Requirements Modeling

The Requirements subprofile contains modeling concepts for describing requirements of the system to be designed. Requirements are typically informal by nature and describe mandatory or desirable properties of the system on a relatively high level of abstraction. In requirement modeling functional characteristics are defined, but also non-functional properties can be specified. Functional requirements specify the behaviour and functionality of the system, i.e. what the system should do. Non-functional requirements, i.e. quality requirements, define the constraints, conditions and prerequisites for the functionality, the implementation and the performance of the

system. Depending on the point of view the requirements may be in a structured detailed form or as textual descriptions.

For smaller control applications with fewer dependencies to related engineering disciplines it is relatively straightforward to specify the requirements for the application according to traditional software development practices. For control of typical industrial processes, however, the preceding and related disciplines contain a lot of engineering information that significantly affects the design and decisions made in control application development. Many of this type of derived requirements contain data that already is or can be easily semi-formally structured. For these the AutomationRequirement concept has been developed.

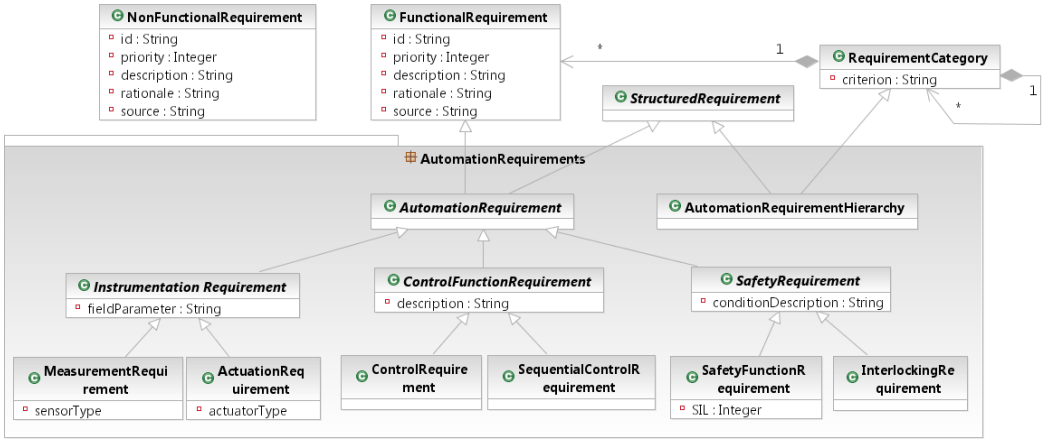

Figure 4 Excerpt from the Requirements subprofile of the UML Automation Profile.

AutomationRequirements (see Figure 4) add additional concepts from the domain of automation and control. In comparison to the other requirement concepts they are also more formally structured with the intention of being processed programmatically. P&IDs and spreadsheet lists of process design, for example, contain automatically processable data such as control functions, connection points for controls and measurements, and the instrumentation. There are specialized requirement types for defining requirements related to instrumentation, control functions as well as safety, and they all have detailed attributes for each purpose.

The concepts have been designed to support the information content in various notations. The requirement model is therefore detailed and also enables describing the requirements explicitly. For this purpose the concept of a StructuredRequirement has been introduced from which the AutomationRequirement also is extended. StructuredRequirements enable complex requirements with more formal structures to be

defined, and allows information such as features and properties to be added in a property-value manner using refinement attributes. The RequirementRefinement is a general concept for refining and specifying additional details to requirements and it is offered in different concrete types as seen in Figure 5. This enables the inclusion of data in almost any format, but, on the other hand, increases the risk of introducing incompatibilities between models if it is not catered for and supported in the development process.

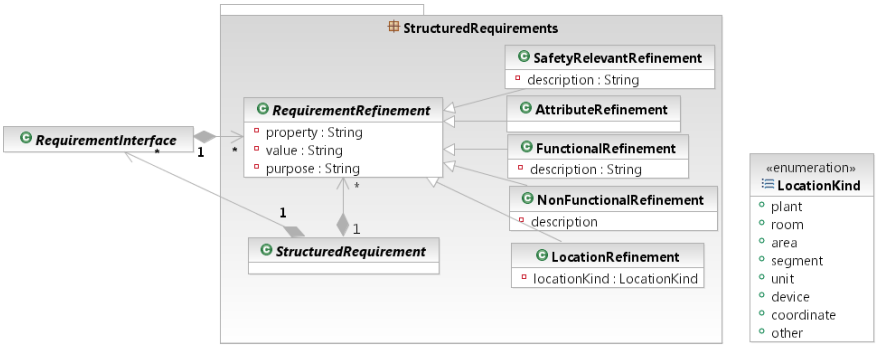

Figure 5 StructuredRequirement is a semi-formal requirement concept of the UML Automation Profile.

The UML AP provides a RequirementRelation concept for modeling relationships between requirements. In addition, a RequirementInterface concept was developed for expressing information exchange dependencies between functionalities in the requirements modeling. For example, using a specialized DataRequirementInterface the required data flow can be specified and using AlarmRequirementInterfaces or InterlockingRequirementInterfaces the information requests can be specified for alarms or interlocks, respectively.

3.2.2 Functional Modeling

The AutomationFunctions package defines the generic base for modeling automation and control domain specific functionality, and the concept is specialized for different purposes as seen in Figure 6. Using AutomationFunctions the structure and functionality of the system is designed to fulfil the requirements. AutomationFunctions can be hierarchical and describe functionality as a network of AutomationFunctions that exchange data and services using so called AutomationFunctionPorts.

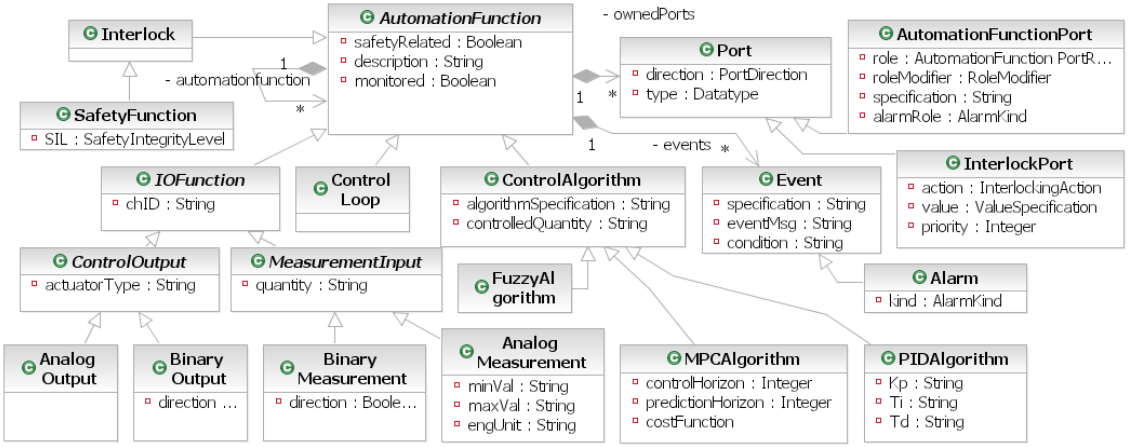

Figure 6 AutomationConcepts are modeling elements used for functional modeling of the system.

The AutomationFunction and AutomationFunctionPort allow modeling of additional design-time properties with a refinement mechanism similar to the one presented for the Requirements subprofile. The AutomationFunctionRefinement is a general concept for defining additional design-time details that cannot be specified with the standard meta-attributes of AutomationFunctions. Refinements of this type specify both the property being defined and the corresponding value for the property. Although the refinements enable support for extended features and customized data modeling, which is sometimes required, the risk to lose important information with incompatible models is increased.

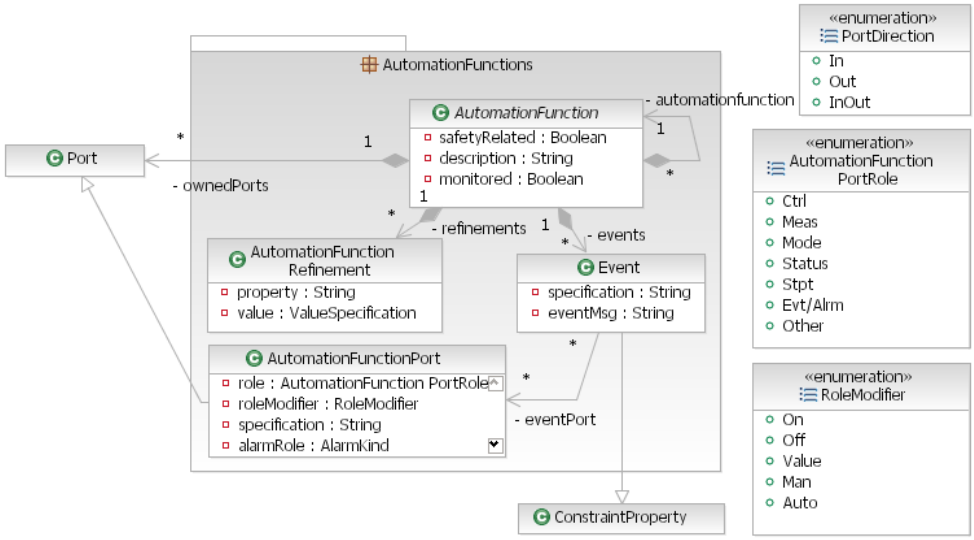

Figure 7 The AutomationFunction concept of UML AP

The AutomationFunctions are divided into measurement, actuation, control and interlocking functionalities. Each type represents a piece of application functionality that connected and in composition forms the desired behaviour of the control system. From this perspective the AutomationFunctions could be described as platform independent type circuits similar to those used by the industry in proprietary DCS platforms. It is important to note, however, that the aim is not to identify the actual type circuits to be used or restrict the selection of possible platforms. The AutomationFunction concept enables shifting design towards platform independent modeling from a traditional PLC or DCS platform specific development. The aim is that for the functional model there are a number of possible execution platforms available that offer several actual type circuits corresponding to each AutomationFunction.

Modeling control functionality on a platform independent level enables developing solutions with domain constructs that can be reused on a number of execution platforms. This is also a benefit for distributed engineering when, for example, the preliminary and basic automation and control design is separated from detailed implementation design and performed by different teams or companies. A platform independent modeling method can also improve the information exchange by introducing a new well-defined modeling layer. This could potentially result in implicit interpretations occurring less frequently as a result from not developing applications using non-normative methods and characteristics of specific target platforms.

3.2.3 Execution Platform Modeling

The existing application platforms of the automation and control domain offer many features, such as reliability, performance, and extensive equipment support, worth supporting in new development approaches or in the transition towards systems of the next generation. In order to utilize existing control system platforms it is required to support modeling the constructs and characteristics of these platforms.

The purpose of the platform dependant modeling phase is to elaborate the platform independent functional model with details of the target platform in order to finally support generation of executable applications. As the platform independent model contains the desired functionality, e.g. abstract type circuits, the elements of the model need to be tied to concrete platform specific elements implementing the functionality. In detail, this means marking the individual AutomationFunctions with specifications of the corresponding implementing constructs and signal interfaces, and specifying the additional required parameters of those elements.

Essentially, the platform dependant modeling relies on platform specific profiles that describe the constructs and characteristics of the execution platform. For example, a platform specific profile is used to model a collection of concrete type circuits on a specific platform. The profiles consist of stereotypes and tagged values that correspond to concrete type circuits and model libraries of complete AutomationFunctions. These AutomationFunctions include the signal interfaces corresponding to those of the implementing elements that as a connected network compose the executable application for that particular platform.

In [P2] the use of a collection of type circuits implemented in IEC 61131-3 FBD [44] for a soft PLC platform is presented. The type circuits represented as function blocks implement, for example, the actual controllers, reading of measurement inputs, and writing of control values to actuators. In a similar manner also proprietary DCS constructs could be modelled as platform specific profiles. However, this has not yet been verified experimentally.

The final execution platform, however, is not limited to PLC or DCS systems, and traditional program code, software components and classes could be used as well to implement the actual features and capabilities of the system. However, in comparison to type circuits and function blocks the transformation is different and more complex. One important reason for this is that the target platform level of abstraction is lower and of a generic nature compared to DCS platforms, for example.

3.2.4 Implementation of the Profile

The UML Automation Profile specification defines how the modeling elements are structured and used to model control software. In order to use the profile and the modeling concepts they need to be implemented and given modeling tool support.

As presented in [118] the implementation of the UML Automation Profile requires meta layer modifications to be implemented. This is due to the fact that the UML super structure specification [77], for example, denies stereotypes from being used to introduce new associations between metaclasses or adding new metaclasses. The UML Automation Profile is therefore based on its own metamodel implementation extending both UML and SysML metamodels. The meta level implementation allows using domain concepts concurrently with standard UML and SysML, and provides greater expandability compared to a DSL modeling approach. This type of profile implementation also enables introducing new custom elements and diagram types. All this emphasizes domain resemblance, the ability to support the needs of the domain

while still providing a semi-formal modeling method that can be used in a MDD paradigm.

The UML AP metamodel is implemented on the Eclipse platform, based on existing metamodel implementations of UML[3] and SysML[4]. For the Eclipse platform there are several UML/SysML tools available that are implemented using the Eclipse Modeling Framework (EMF[5]). EMF also provides good support for various model to model transformations, and a number of tools and technologies, such as QVT [71] and ATL [48], are available. The metamodel implementation, and the tool support in general, has been described in detail in [118].

The implementation of the profile metamodel on the Eclipse platform restricts the usage to that environment, but considering the available options it is among the most open architectures and based on an active open source community. The control application models developed using UML AP are therefore dependant on the Eclipse environment and the extended plug-ins. From a distributed engineering point of view this requires using a tool environment supporting this implementation. For transfer of models and application semantics the XML Metadata Interchange (XMI) [73] is used to serialize the models. XMI enables exchanging metadata information of UML and SysML, and in principle any other MOF based modeling language, using an XML based markup. The aim is to be able to transfer the abstract model semantics between different modeling tools but because many of the XMI implementations of different vendors are incompatible this is not always possible. The XMI specification has not redeemed its full expectations, and the standard has been shown defective in supporting model exchange between different tools for example by [59] and [86]. For UML AP models it must be noted that the tools would additionally have to support the concepts both syntactically and diagrammatically, i.e. have a metamodel implementation, in order for the full information exchange to be possible.

[3] org.eclipse.uml2

[4] org.topcased.sysml

[5] Eclipse Modeling Framework Project (EMF), URL: http://www.eclipse.org/modeling/emf

3.3 Complementing MOF Based Metamodeling with Ontologies

Models in all areas of development are becoming more complex as more features are being described. Models of different disciplines are also increasingly being linked which further makes management of models challenging. Methods such as Semantic Web ontologies could be used to improve knowledge management of models.

Another interesting use case for the semantic descriptions of control applications is within composition of functionality and features offered by a system. It has been proposed, for example, that semantic descriptions can be generated from traditional UML models [1][37] or created for manufacturing systems [90] in order to be used automatically in composing functionality.

Modeling constructs being used to describe control applications, for example, include many hidden semantic issues that are apparent to the human designer but not interpretable to automatic model processing tools. The proper utilization of a construct can also be limited for a designer if the implication of some concept is not known and understood. Even if some meta level information is in a machine interpretable format the semantics are usually limited to the particular modeling language and tool environment which prohibits more extensive management and reuse of domain knowledge.

An important issue is also the capabilities of e.g. a metamodel to describe and include additional meta level information related to semantics of modeling language constructs. The reality often is that the modeling constructs can be enriched with semantics using mechanisms and constructs other than those of the metamodel. A main reason for this is the design purpose of the metamodel, i.e. the role of specifying modeling language constructs and defining the syntax of their use.

For this two approaches are presented that allow the use of OWL ontologies and Semantic Web technologies to be used as a supplement to traditional modeling methods. The first approach, described in [P3], is focused on utilizing development environment independent solutions e.g. in distributed engineering networks. The second approach, presented in [P4], strives for better performance and concurrent utilization in an IDE setting. Characteristic to both approaches is division of engineering information into domain and instance knowledge, and separation of both knowledge transformation processes, as presented in section 3.3.2.

3.3.1 Differing Paradigms

Metamodel based modeling and ontology development differs in many ways. Compared to a metamodel based model an ontology allows significantly more freedom in expressivity and combining of other, previously defined knowledge. The ontology modeling approach, and the Semantic Web, is designed flexible, extensible and less restricting compared to object-oriented modeling. In [80] object-oriented programming has been characterized stating that objects must be a member of exactly one class, inherit only one superclass, and exactly conform to the structure of the class specification. This is analogous also for typical modeling languages, based on the object-oriented paradigm.

Semantic Web ontologies are not restricted by a metamodel in a semantic sense, i.e. there are no restrictions on what can be described. Descriptions such as RDF and OWL ontologies can have multiple types and superclasses. More importantly the ontology individuals can differ from these intended definitions or not have any definitions at all. The definitions are used to identify individuals belonging to a certain class based on the properties of the individuals.

3.3.2 Approach Separating Domain Knowledge and Instance Data

In [P3] a method is presented for using OWL ontology descriptions in addition to metamodel based control application models. The method is based on transforming model information to ontology descriptions as depicted in Figure 8. Distinct to this approach is a clear division of domain and instance knowledge. The former refers to information of a static kind, i.e. the metamodel and other semantics related to the modeling method. The latter, on the other hand, contains only the structures and data of the instance model, e.g. a control application designed for a specific purpose. This separation similar to metamodels and model instances for e.g. UML modeling is not as definite for Semantic Web ontologies. The approach allows keeping model instance transformations simple and efficient [P3].

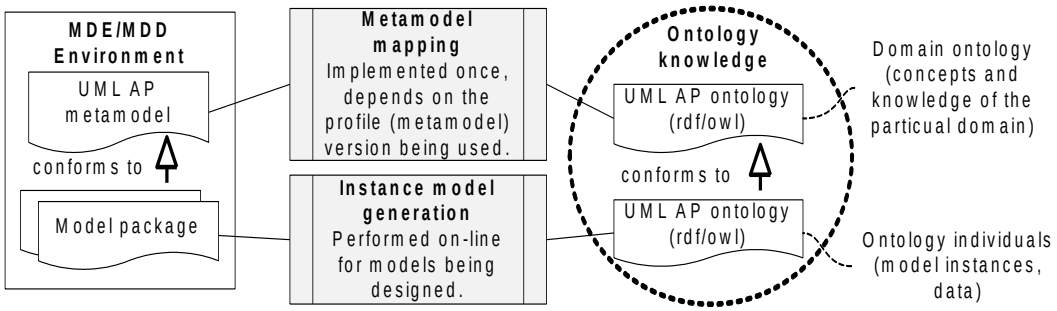

Figure 8 The transformation of instance data is separated from domain knowledge management [P3].

According to MOF definitions of metamodel levels the UML AP implementation on the Eclipse platform is on the M2 layer. This requires metameta level constructus to be used in the transformation generating a domain ontology corresponding to the modeling language. Figure 9 illustrates the process for UML AP when the metamodel constructs are transformed into an OWL ontology according to the MDE model transformation pattern [48]. This ontology can then be used to define new modeling element semantics, and for reasoning purposes to further infer new knowledge.

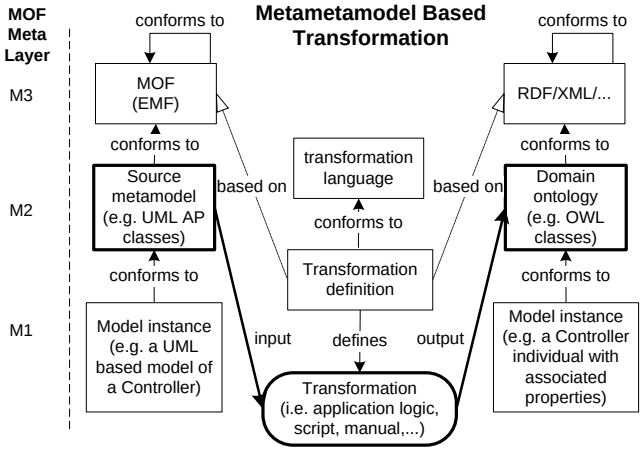

Figure 9 The domain ontology creation can be seen as a metametamodel based model transformation of the domain modeling constructs. [P3]

The UML AP models of the design of the engineers are, on the other hand, on the M1 level. Transforming these model instances to corresponding ontology individuals can be considered as model to model transformation based on metamodel definitions, i.e. a transformation definition of how each modeling language construct should be transformed into a corresponding domain ontology concept (see Figure 10). Due to this separation of knowledge the instance model transformation mitigates to iterating model structures, and populating and connecting respective ontology individuals.

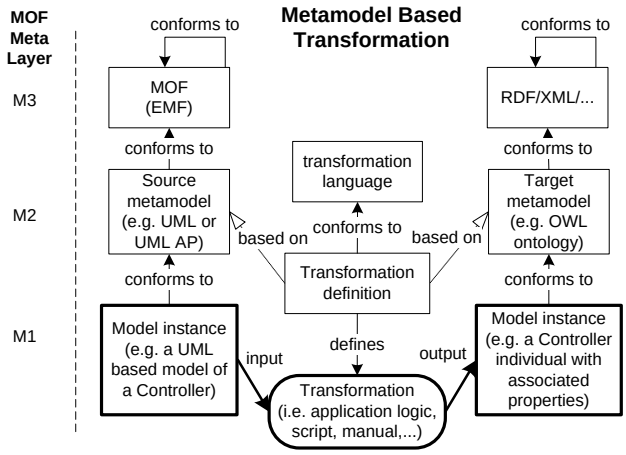

Figure 10 The instance ontology transformation is based on populating properties to individuals defined by the domain ontology. [P3]

The two transformations differ in a constitutive way considering implementation. First of all, metametamodel based transformation of modeling languages, e.g. UML AP concepts to domain ontology concepts, can be complex as the defining elements from the M3 level are very generic. This may even result in the fact that automatic transformation rules based on metametamodel concepts are impossible to implement. Secondly, the need of such an automatic transformation is also debatable because modeling languages are usually standardized and established. For DSLs, however, an automatic transformation is preferred for example when the modeling language evolves in pace with the development. The transformation of model instances, on the other hand, should be automated in order not to slow down development or introduce new risks of manual errors.

Because the UML AP metamodel is based on UML and SysML also respective domain ontologies are required if meta information of those modeling languages is to be supported. This means that to manage meta information of all involved model element types also those modeling languages need to be transformed to domain ontologies in order to process the instance models with UML or SysML concepts. In the case of UML AP this is, however, not necessary as most of the concepts have their own semantics and the UML and SysML background is mainly utilized in building the modeling tool support.

3.3.3 Development Environment Independent Instance Transformation

The model instance transformation initially presented in [P3] has been developed to support transformation of UML AP models to OWL 2 ontology individuals with the aim to minimize dependencies to the modeling tool environment. The purpose of this is to allow development to be managed in engineering processes typical to distributed engineering, and allow integration of additional engineering related information being used.

The Eclipse and EMF based metamodel implementation allows exporting the UML AP models to XMI. Despite of the XMI standard, as explained in section 3.2.4, this means that a general class based transformation is not available and a specific adapter is required for accessing the model contents in transformations. Also as a result of using XMI the graphical diagram information is not transferred to the ontology descriptions, and any additional semantics are limited to the syntactical source model.

With model data in XMI, and knowing the format, a XSL transformation can be developed to process the XML based source document. As a result the transformation generates corresponding OWL individuals by using the XML based OWL serialization as the transformation target. The transformation is illustrated with an excerpt of comments in Figure 11.

Considering the XSLT implementation it must be noted that the transformation is not a genuine metamodel based implementation. It lacks, for example, a strongly typed metamodel and utilization of class inheritance in transformation definitions. This means that some transformations for similar objects have to be redefined which increases redundancy and makes maintenance of the transformation logic more challenging. This is not a major concern as many of the processing features are generic by nature due to the simplicity of iterating model constructs and generating corresponding OWL individuals.

Converting UML AP models to OWL ontologies using the approach is relatively fast even considering models including hundreds or thousands of objects. The method has been successfully utilized in [P6] to connect auxiliary information and analysis to engineering processes outside the modeling environment. In an IDE, however, a transformation delay of even a second or two results in the method not being practical for providing timely supporting information, for example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" ... >
    <xsl:output method="xml" version="1.0" encoding="UTF-8"/>
    <!-- General variables and templates -->
        <!-- Namespace URIs and variables,
        expressions for selecting different objects -->
        <!-- Template for defining unique names -->
        <!-- Template for port connections between individuals
        (performance compared to rule based inferance) -->
        <!-- Template to assert traces from requirements to functions
        (performance advantage compared to rule based inference) -->
    <!-- Instance ontology transformation template -->
    <xsl:template match="/">
        <Ontology xmlns="http://www.w3.org/2002/07/owl#" ... >
        <!-- Import UML AP domain ontology -->
        <!-- Declare named individuals of UML AP objects -->
        <!-- Class assertions for the UML AP objects -->
        <!-- Objectproperty assertions for UML AP individuals,
        i.e. connections to other objects -->
        <!-- Data property assertions for UML AP individuals,
        i.e. attributes of elements. -->
        <!-- Port connections -->
        <!-- Trace relation connection -->
        <!-- Additional assertions of global kind -->
        <!-- Declare the sourceDomainId data property functional to
        distinquish different individuals -->
        </Ontology>
    </xsl:template>
</xsl:stylesheet>
```

Figure 11 Excerpt of comments presenting the structure of the XSLT document for transforming UML AP XMI serialized models to OWL 2 XML individuals.

3.3.4 Incremental Transformation for IDE Integration

The transformation approach targeted for IDE integration aims for better performance in order to enable semantic descriptions of the model to be used concurrently in real-time [P4]. The method is based on simultaneously maintaining an OWL 2 based knowledge base of the UML AP model being designed.

Figure 12 illustrates the architecture for the IDE integrated transformation approach. UML AP Tool Editor is a part of the Eclipse and EMF based tool support, presented briefly in section 4.3, developed for modeling using UML AP concepts. The Semantic KB (SKB) is the component that concurrently maintains an OWL representation of the

model being designed. The SKB contains an OWLAPI[6] based OWL knowledge representation, and utilizes a UML AP to OWL transformer component for updating the OWL based descriptions.

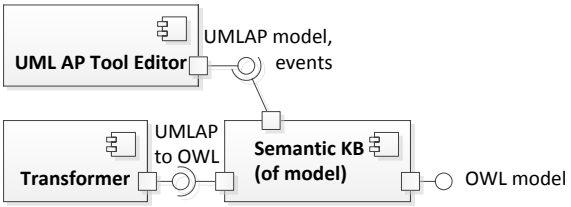

Figure 12 The Semantic knowledge base maintains OWL descriptions concurrently with the UML AP model in the editor. A transformer component is used to address occurring changes in the UML AP model and incrementally update the knowledge base accordingly.

UML AP Tool Editor enables events to be monitored and reacted upon when selections or modifications to the UML AP model are being done. It also enables access to iterate and access the EMF based UML AP model tree. When events of interest, e.g. selections, additions or removals, are registered by the SKB it invokes the transformer to update the knowledge base with a reference to the UML AP model, the altered element, and the OWL knowledge base. The transformer directly accesses the models without the need for serializing and deserializing models, and performs an incremental update to the knowledge base. Adding or modifying UML AP elements results in the transformer to redeclare the corresponding OWL individual from the ground up. This includes declaring all the property axioms of the individual as well as the property axioms of other individuals the altered element are be part of. When an UML AP element is removed all the axioms in the knowledge base it is part of are simply removed to maintain consistency of the knowledge base.

Considering performance the simultaneously maintained semantic knowledge base caused no noticeable overhead in processing the models [P4]. In brief experiments typical increments to the knowledge base were measured in the range of milliseconds to tens of milliseconds on a regular desktop computer. The IDE integrated transformation is utilized and demonstrated in various on-line work support tasks in section 4.4.

[6] A Java API and reference implementation for creating, manipulating and serializing OWL Ontologies. URL: http://owlapi.sourcegorge.net

3.4 Discussion

UML AP provides a semi-formal modeling method that includes structured elements for supporting the control software development process. The modeling constructs are designed with domain-specific requirements in mind and the elements strive to provide resemblance to concepts familiar to professional designers. The resemblance to the extended modeling methods, namely UML and SysML, is not apparent. During an organized evaluation event with industry professionals, referred to in [P2], it was pointed out that designers should not need to understand UML nor SysML concepts in order to be able to use the profile concepts. It was also concluded that the concepts are familiar enough for learning to use the method as any other tool used in engineering.

In comparison to DSL approaches the UML AP modeling approach is less restricted considering the modeling method. UML AP allows using both UML and SysML constructs concurrently with domain concepts in the same models. This is of significant value as industrial control software is often connected to other information systems or includes components that cannot be described using a limited set of domain concepts. In cases where DSLs typically are applied the problem space is, on the contrary, more restricted and a limited set of solution concepts is sufficient.

Experience has shown, regarding utilization of existing source information, that the modeling language must not be too restrictive to support existing practices. For this reason UML AP includes several types of requirements that allow different aspects of preceding or related engineering to be integrated and modelled either as freeform textual requirements or as detailed requirements containing structured data. The requirement modeling then allows requirements to be connected with relations, and connecting the requirements to implementing functions to ensure quality and traceability of design decisions.

UML AP provides a common means for specifying the functionality and features of the application using domain-specific concepts instead of generic implementation level constructs. This contributes to increasing the level of abstraction, and enables better utilization of the expertise of domain professionals to implement the application instead of e.g. software specialists. Although not measured, it is also believed that describing functionality and characteristics of the system using domain concepts can have a positive effect on quality and reduce the possibility of errors. In addition, an implementation independent modeling approach makes way for future solution reuse among a number of possible implementation platforms.

UML AP supports existing platforms and provides means for modeling their components and features so that they can be used as building blocks for the platform specific models. The platform modeling has been successfully verified for a collection of type circuit function blocks on a soft PLC platform. The modeling has not been experimentally verified for a DCS platform, but there are no apparent limitations known that would prevent it from modeling basic functionalities. During the aforementioned evaluation event the interviewees also indicated that constructs of commercial control systems could be modelled similarly.

Concerning OWL there are many drawbacks and limitations why it as a method is not appropriate to be used as the only modeling method for control applications. The most apparent from a designer's development perspective is the lack of a graphical model that would have resemblance to the domain concepts as is the case of UML AP, for instance. OWL and typical Semantic Web ontology technologies also lack syntactical enforcement of restrictions and conditions, and are free from such constraints that limit the kinds of knowledge to be expressed. They also lack means for evaluating omissions in semantic descriptions, and many features familiar in metamodels are not available to assure integrity.

Adopting OWL ontologies is proposed in this thesis as a supplement to application models to describe additional features of the model, and to enable integration of other engineering knowledge. The metamodel of the modeling language ensures that the model conforms to the rules and practices of how constructs are used. The additional semantic layer opens up new possibilities to integrate and manage additional meta level information that can be used to support design.

A special characteristic of the method proposed is the distinct separation of the modeling language and the actual models developed. Considering the ontology representations, a domain ontology and an instance ontology is prepared to correspond to the modeling language constructs and the instance model, respectively. By dividing the ontology knowledge extraction into two different tasks they can be managed and executed separately. On one hand, this results in the instance transformation to remain relatively simple and straightforward to implement as only the model objects have to be transformed to ontology individuals with corresponding properties. The domain ontology, on the other hand, is in many cases, including UML AP, updated less frequently and an automated transformation may not be necessary. This, however, depends on the metamodel update frequency which in development of a modeling

language or in the case of DSLs results in additional maintenance required for the domain ontology.

For transforming model instances to ontology individuals, two different but compatible methods were proposed for producing corresponding OWL ontology descriptions of application models being designed. The first transformation method offers environment independence while the incremental transformation in the second one is designed to be implemented and integrated in modeling tools. Although the first approach is relatively efficient considering the size of the models it cannot compete in performance with the incremental approach designed to offer concurrent OWL descriptions of the models being designed.

OWL ontologies of control application models are not only another representation form of the modeling language and the model being designed. Although the semantic descriptions allow various reasoning and inference mechanisms to be added they do not provide absolute value by themselves unless additional semantics for this is used. In order for the method to pay off the semantic descriptions need to be extended or combined with other existing ontologies.

4 Engineering Methods for Industrial Control Applications

Control application development is a complex engineering activity that interacts and is dependent on related engineering disciplines. The process involves communicating with, for example, process engineering, mechanical design, equipment manufacturers, instrumentation, network design, electrification, instrumentation, and security and safety design. Multidisciplinary knowledge is often required to develop applications that control the processes using the intended equipment as planned.

Development methods are required that support this communication and allow requirements from different stakeholders to be taken into consideration. As discussed in the previous chapter, the tools and information models are heterogeneous, and sophisticated import and export capabilities are essential in handling the large amounts of data. The functionality of control applications has also increased thus requiring means to efficiently handle model content using concepts appropriate for the task at hand. The large number of different control system platforms also calls for new methods that promote reuse of designs and well-proven solutions.

The use of general UML in combination with IEC 61131 execution platforms has been proposed and studied by [13, 19, 57, 91, 100, 129] among others. The use of UML based development methods has also been studied for IEC 61499 platforms [24, 29, 40, 83, 111, 121]. Composable Automation Components, combining software and hardware, has been presented as a solution to closer integration of engineering and system development [108, 109]. The referred approaches among others are detailed in [P2].

It has been argued that the use of ontologies throughout the software development lifecycle could bring enhancements from the logic-based formalisms and semantics [31]. It has been stated that when the context space is expanded also the applications need to be more intelligent and this can be supported with development methods employing formalized concepts [103]. The interest to use ontologies in different application areas has increased especially concerning knowledge management [15]. Examples of relevance for control application development are e.g. formal application model semantics for model-driven engineering purposes [106], interoperability of models in design and formal safety verification analysis [62], and querying UML designs for reuse [95]. Integrated use of UML class based models and OWL ontologies have also been proposed, and a framework has been developed to support this type of

engineering [84]. The referenced approaches, however, more or less involve the engineers to use both OWL and traditional modeling methods in their work.

This chapter discusses development of control applications using model-driven methods and defines requirements for implementing development processes. For this a model-driven approach is considered based on the modeling concepts presented in section 3.2. The OWL based semantic extensions proposed in section 3.3 are also utilized to automatically support different tasks in control application engineering.

4.1 Model-driven Engineering of Industrial Control Applications

4.1.1 Domain-specific Modeling

Models have been often used in industrial automation and control to describe and analyse systems. Models, however, have typically been used only to describe structural aspects of the systems on a relatively high level of abstraction, or to model the behaviour of the system dynamics using mathematical models for regulatory purposes as discussed in [114] or [38], for example.

One of the primary motivations for model-driven techniques is to allow designers to concentrate on aspects of application development that genuinely bring value to the solution [P2]. This often means using building blocks of the problem space, i.e. domain-specific constructs, which typically are on a higher level of abstraction than traditional implementation technologies such as programming languages.

This is especially beneficial, and worthwhile in spite of possible modeling overhead, for larger applications including thousands of design objects. Model-driven techniques can automate transitions between modeling phases and reduce the amount of routine work otherwise required [12][117]. Allowing design to focus on key issues instead of implementation details can be seen as a prerequisite and key component in achieving a higher degree of quality. This is desirable for engineering of large industrial control applications that connect a great number of engineering objects as a network. Industrial scale DCS systems can be composed of thousands of measurement and actuation points from a multitude of devices with different hardware and software configurations [50].

4.1.2 Domain Requirements for a Model-driven Approach

Requirements for applying model-driven methodologies to development of software for the automation and control domain can be derived from the nature and practices of the industrial control domain and the strains of involved multidisciplinary engineering, as discussed in [P1] and [P2]. The requirements also serve as a motivation of adopting and achieving the promises of a model-driven methodology.

- Support for multidisciplinary engineering input - As the development of control applications is influenced by related engineering disciplines the requirements and concerns of those are necessary to be included in the development process. In addition to enabling existing engineering information to be integrated the development method should allow follow-up of this information to developed solutions.
- Modeling concepts of the domain for different phases of development - The development method offers suitable constructs for each phase of development. This requires modeling constructs of various abstraction levels that are familiar to the professionals and improve communication of designs between developers within the domain and closely related engineering disciplines.
- Improved quality and reduced possibility of errors - The model-driven development philosophy relies on a modeling method that typically includes a limited set of modeling constructs, and adheres to a defined method of organizing and connecting constructs. For example, compared to traditional programming languages or general modeling the solution space is deliberately restricted thus reducing the number of different ways to implement the intended functionality.
- Scalability and management of large models - Typical targets for control applications can include tens of thousands of objects related to the plant and it is not uncommon for the control application to be related to thousands of points of reference or input and output connections of the hardware equipment. A development method therefore needs to support logical division of model components for efficient management during the development process.
- Efficiency and increased productivity of application development - Application engineering often includes many error-prone transfers of data sets and listings that should be automated and programmatically assisted to reduce manual routine labour. The purpose of the control application is to automatically

monitor and control a process according to specified criteria. The main task of the control application engineer should be developing this application functionality without having to focus too much on implementation level details and technologies.

- Improve reuse of solutions - The method should support typical domain reuse of well-tested solutions, e.g. recurring functionality such as control of certain types of electrical motors, and also allow reusing parts of or whole developed control applications for different implementation platforms.
- Distributed engineering - Engineering work is distributed among teams that even span between enterprises. The development approach should provide well-defined points of interaction and support design by contract to ease management and transfer of information in distributed engineering.
- Support developing applications for various hardware and implementation platforms - The development method should not steer the development unnecessarily toward a specific implementation platform. Due to the existence of a multitude of platforms used by the industry the method should also not unnecessarily limit the use of well-tested and profound implementation platforms.
- Extensible to allow modeling of heterogeneous and different types of systems - Although the modeling paradigm should be based on profound concepts the methods should allow modeling of different types of functionality and varying structures, as well as integrations to other information systems.

4.2 AUKOTON Development Approach

During the AUKOTON project a development approach consisting of three distinct phases was developed and outlined in [P1] (see Figure 13). In the first phase of the control software development requirements are specified along with gathering relevant data from preceding and relevant engineering disciplines. The requirements are elaborated and organized to serve as the starting point for the following functional design phase. At this stage of development the functionality of the application is specified in a platform independent manner, and the application architecture and distribution of components is determined. In the final third stage the platform independent functional model is bound to existing components of the implementation platform. The building blocks of the development approach are the concepts and

existing practices of the industrial control domain, and they are implemented as the UML Automation Profile (presented in section 3.2).

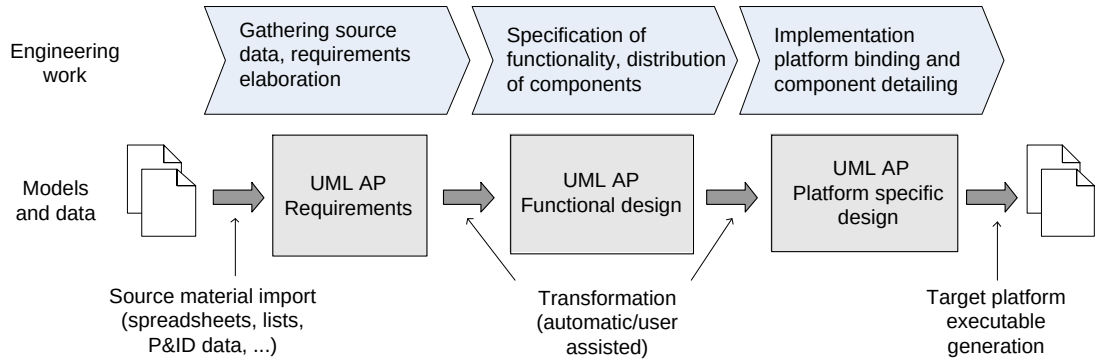

Figure 13 The model-driven AUKOTON development approach.

In [P2] a development example of a control application is presented for regulating a water process. The same example process and control application has also been used as a part of the organized evaluation event for industry professionals participating in the AUKOTON project. The arrangements of the evaluation event are described in more detail in [119].

4.2.1 Requirements Import and Requirement Modeling

In the example, data is first imported from spreadsheet listings of preceding process design in order to establish the initial requirements model. The first imported file contains I/O points for measurements and control along with a number of requirement attributes needed in instrumentation and parameterization of the equipment. The second imported file contains safety related interlocking functionality requirements for interlocks that are to be implemented as a part of the control application, e.g. by using the I/O points of the first imported data set.

The use of IEC 62424 [42] and CAEX XML as viable control application input has also been evaluated in a similar fashion as a source for structured requirement data that can be imported automatically or in a user assisted manner. The UML AP requirement modeling constructs have been designed to cover this standard data related to control functions as this also represents typical background information of the domain.

The imported data serves as a starting point for elaborating the requirements model. Despite the fact that most of the imported data can be automatically processed to requirements, and the process can be further improved with user assisted import work

flow, there is genuine requirement elaboration work needed in this phase of development. As the goal of the requirements modeling phase is to gather all requirements affecting the control software development, the requirements derived from preceding and concurrent engineering require modifications and restructuring to cater for the application development point of view. New requirements concerning the control application software, e.g. platform requirements, are also introduced, and the resulting requirements model is used as the foundation guiding the following design and implementation of the control application. This also includes specification of traditional textual requirements of both functional and non-functional kind.

4.2.2 Functional Platform Independent Application Modeling

The elaborated requirements model contains requirements of different types and categories, e.g. measuring or controlling certain process values. The AUKOTON development approach has been designed to utilize the requirement model extensively, and based on this an initial functional model can be automatically generated as a basis for the functional modeling.

Especially StructuredRequirements (see section 3.2.1) are of value as they include information on what kind of functionality is required and what type of AutomationFunction (see section 3.2.2) could be used for designing the implementation. In the control application example of [P2] the result is a skeleton for the functional model that is created based on the StructuredRequirements. The functional model needs to be further developed with various details, organization of functionality, intermediary logic, and taking into consideration those requirements that could not be automatically transformed into the functional model.

The functional modeling is where the most significant part of the work in the development process is performed. The purpose is to design the control application using specialized UML AP AutomationFunction concepts that could be characterized as abstract type circuits or function blocks typically available on proprietary DCS and PLC platforms. These constructs represent functionality for measuring process values, performing control computations, and controlling actuators and equipment of the process. At this stage of development the designer also has to take a stand on how functionality is distributed and how information is transferred between the functions.

4.2.3 Platform Specific Design

The third phase of the development approach consists of choosing the execution platform, and which suitable constructs, i.e. type circuits and functions blocks, are to be used to provide the requested functionality of the functional model. For this the functional model of the previous phase is used as a starting point to which tags of concrete type circuits and function blocks are added. Typically the execution platform also requires specific input of details related to the capabilities and limitations of the chosen platform.

In order to be able to choose suitable concrete constructs of the execution platform the constructs need to have been modeled previously as explained in section 3.2.3. The process of choosing corresponding and sufficient implementation constructs can be assisted in the IDE based on metadata of the functional model and the previously modeled execution platform constructs. As a result of this phase a UML AP based model is created that describes the final control application software to be run on the chosen platform. This model is then ultimately used to automatically generate the executable code or binaries for this execution environment.

The aim is, however, that there doesn't have to be an execution platform readily available and planned, and that the same functional model, or a part of it, can be reused on another implementation platform. It is typical to the domain that functional design is performed by one company or a team and the implementation design, including choosing the platform, is carried out by another stakeholder.

4.2.4 Model Transformations and the UML AP Metamodel

The AUKOTON development approach has been designed to use suitable modeling constructs for different needs, and with model transformations in mind, to support the transition between models in different situations and phases of development. UML AP defines its own metamodel (see section 3.2.4) and most of the model transformations and information processing between the models is based on the metamodel.

Typical source information for requirements import does not usually adhere to a modeling formalism based on a metamodel. Nevertheless, metamodel transformation techniques have been successfully applied also for importing requirements by constructing a metamodel of the information content to be imported, and defining the transformation rules between the source metamodel and the UML AP metamodel [117]. For typical source information import, e.g. spreadsheet listings, the metamodel has been

used as a guideline for writing the import logic that populates the requirements model from the source files.

The requirements modeling phase unifies source information to be further used in the AUKOTON chain of development. When the requirements unification is completed the information content is defined according to a well-defined abstract syntax, i.e. the UML AP metamodel. The metamodel enables one to create tools and mechanisms for assuring and testing conformance of models, and also to contain additional semantics of models. It is this meta information of actual models that is especially valuable when defining transformations in a generic way based on the metamodel characteristics. Considering the AUKOTON development approach it is due to the metamodel that the structured, semi-formal information content in requirement models can be automatically transformed to the next phase of development. Due to the differing viewpoints the transformations between UML AP concepts are not always one-to-one mappings as the level of abstraction is changed.

When the design continues from functional modeling to a specific platform the UML AP metamodel is also of great importance. Applying features from platform profiles relies on the previously modelled constructs of the chosen platform and how they can be applied to various constructs of UML AP. Although the platform profiles are merely external interface models of type circuits and function blocks they are important in the generation of the final executable application. These platform profile models have their own semantics, originating from the implementation platform which in turn guide the generation of an executable for that platform. The source model contains all the necessary information and should be unambiguous from the execution platform point of view. The final target transformation object does not have to be based on a metamodeling paradigm and application code or binaries could be automatically generated based on the source model but other types of platforms have not yet been tested [P2]. In principle, the approach could be used to model even the lowest level of constructs of a platform, and to build intermediary models to be used in the transformation of functional models and in the application of platform specific features.

4.2.5 Evaluation of the Development Approach

Quality evaluation for model-driven engineering methodologies for Web applications has been studied by [22], and a number of useful quality characteristics have been identified that are also applicable to model-driven development in general. According to this study the key characteristics to be evaluated in a methodology are usability,

functionality, portability, reliability, and maintainability. These are of importance also for control application development but with a different emphasis, especially for the sub characteristics, due to the differing nature of the domain.

Regarding this thesis and the topic of distributed engineering the most important characteristics from a usability perspective are understandability and interpretability of designs. To improve these characteristics the modeling concepts used in the development approach are based on concepts familiar to the industrial control domain. Combined with a controlled method and a restricted set of constructs the possibility of error and misunderstanding can be reduced. The familiarity of concepts also contributes to reducing the risk for errors in misunderstanding concepts and in interpreting communicated designs which are important concerns in the often mission critical nature of the systems being developed. The familiarity of concepts from the point of view of development method guidance has been further discussed in [92].

Concerning functionality it is especially accuracy that, in terms of providing the sufficient amount of information as a result of engineering, strikes as an important issue in distributed engineering. Designers in different teams integrating and using existing models as input need to have all the necessary information available. The development approach furthers this by supporting consolidation of source information, and offering well-defined modeling formalism to design the application logic and further communicate designs unambiguously. This is also enhanced with trace relations that track back decisions made in the development between parts of the model.

The metamodel based UML AP implementation enhances following of agreed practices and assists in assuring compliance of models. Concerning transformability the metamodel provides a solid foundation for defining and executing model transformations. In addition to transforming models within UML AP this also improves transformation capabilities for other standardized notations and information models.

Maintainability of the UML AP models in the development process depends primarily on the implementation of the models that in turn affects transportability and programmatically accessing and editing data. The metamodel implementation is based on EMF and other open source software and the models are therefore open and the use is not restricted, e.g. by commercial licences. Despite model serializations using XMI there is no notable compatibility between different modeling tools because of divergent uses of XMI and, more importantly, due to the custom UML AP concepts not supported in other tools. For maintainability it is also important to be able to analyse and test the models. The modeling foundation enables analysis to some degree using Object

Constrain Language (OCL, [69]) and the metamodel has also proved sufficient in generation of simulation models for evaluating the design in early phases of development [116].

From a reliability perspective important characteristics are maturity and fault tolerance in case of errors. It is recognized that the profile will evolve in the future with new concepts which may temporarily reduce interoperability of models and reliability of designs. The less restricted property like of refinement elements are seen necessary as a part of this evolution but they may also further scale down compatibility and maintainability of information content described using those elements. Concerning fault tolerance no additional mechanisms have been developed to UML AP or the AUKOTON development approach beyond what the metamodel by itself restricts or what the trace relations offer in terms of backtracking engineering decisions.

4.3 Tool Environment for Model-driven Engineering

The tool environment, UML AP Tool, is not a contribution of this thesis although the work has contributed to the specification and implementation of the tool environment and several plug-in components. The tool is presented as background information related to the implementation and prototyping of the model-driven development approach. The environment is also used as a prototype platform for the semantics enhanced development in section 4.4 as well as a source of models in the service based engineering process management discussed in chapter 5.

4.3.1 Introduction to UML AP Tool

The initial goal for the UML AP Tool was to provide modeling support for UML AP. Eclipse and EMF was chosen as the implementation platform due to open source licencing and the solid foundation it provided for building custom extensions. In the implementation UML AP Tool extends the Topcased [26] modeling tools for UML and SysML.

The tool offers views for navigating the model hierarchy and graphically editing UML AP model elements in different types of diagrams (see Figure 14). The tool also includes a number of plug-ins that implement a graphical user interface for interacting with the designer. These types of tools can be launched from the diagrams or the model tree to assist in imports or transformations, for example. The tool implementation is described in detail in [118].

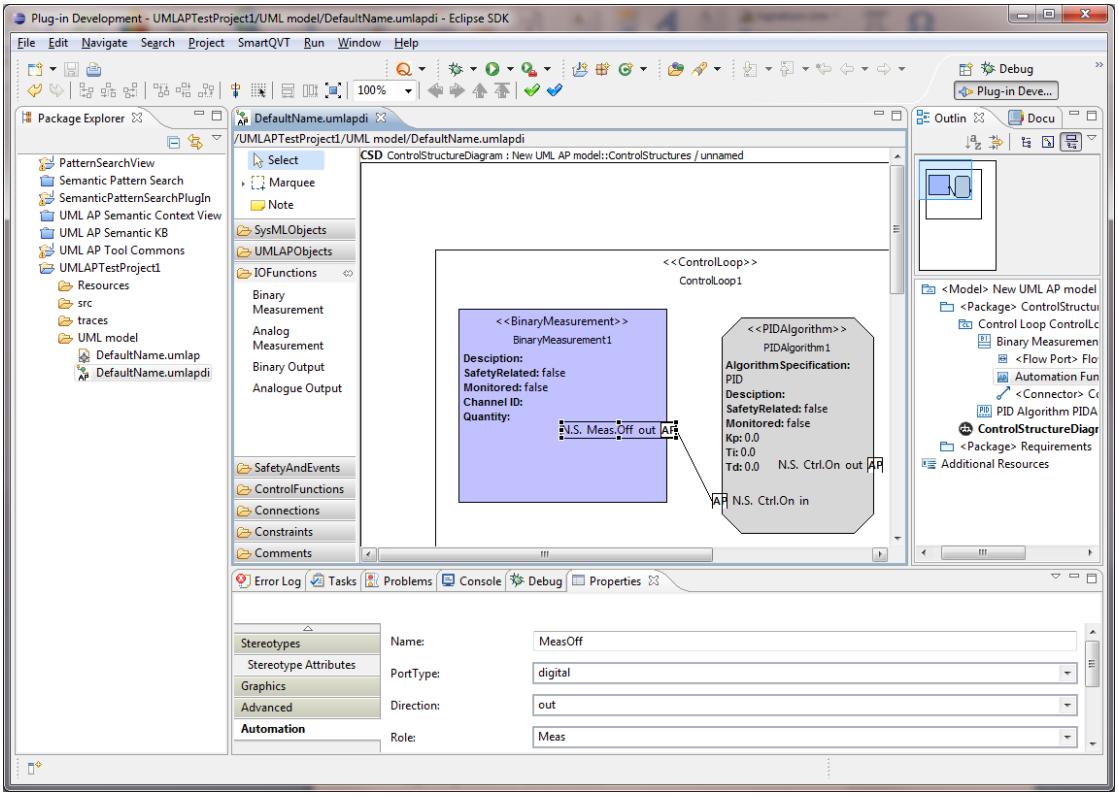

Figure 14 UML AP Tool: Editing a model in a Control Structure Diagram

4.3.2 Extensible Plug-in Based Tool Architecture

The UML AP Tool is based on the plug-in architecture of Eclipse which enables new functionality to be developed and deployed in the IDE. New plug-ins can be developed that utilize functions of existing plug-ins, e.g. to support new transformations or other type of adaptation, additional features, or supporting tools that assist the development.

The tool has been further developed to support the model-driven process of engineering control applications. For this a number of plug-ins have been created that support design and automate model processing with various transformations in the model-driven chain of development (see Figure 15). A few import plug-ins have been developed that can automatically process source information and populate the first model in the development, i.e. the requirements model. Similar export plug-ins have also been created that in turn produce executables to be run on implementation platforms. The extendable plug-in architecture for transformations is described in more detail in [117].

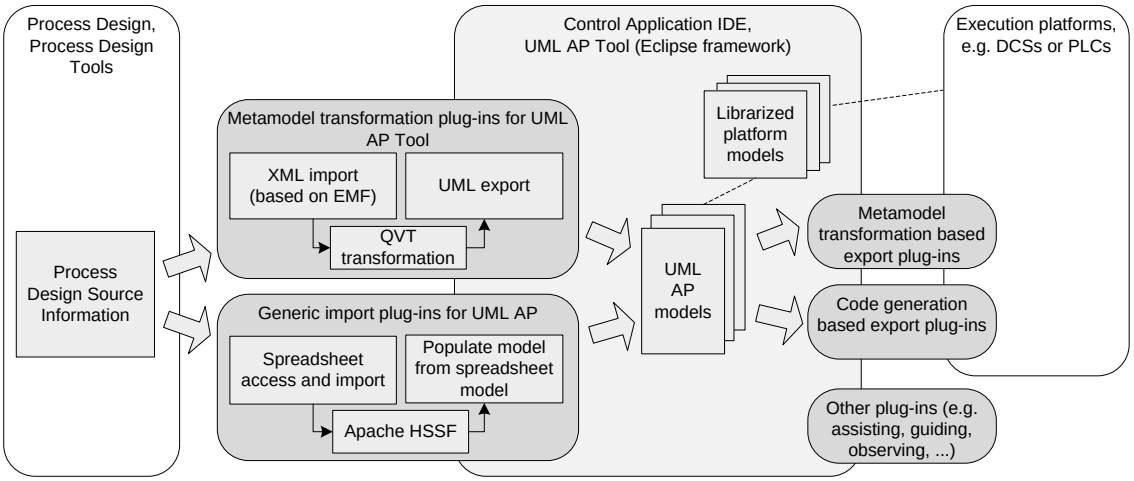

Figure 15 Programmatic data import of process design information to control application development.

Also other types of plug-ins have been developed in addition to the transformation and import/export related ones. For example, plug-ins that give guidance for using the development method or support choosing correct elements can facilitate the proper use of the development approach. Plug-ins that observe the user actions, on the other hand, have been developed to enable creating interactive tools in the future that assist the developer in a context aware sense.

4.4 Semantics in Engineering

The use of new abstractions such as modeling languages to represent and describe functionality causes new challenges. Many of these challenges are related to supporting the use of these new constructs, dealing with partially overlapping viewpoints, using the constructs in model transformations, and maintaining traceability in models [28]. The developments in information and control systems over the decades have also resulted in the systems becoming larger and more complex, and requiring broader expertise from a multitude of disciplines. The respective engineering work is also performed in collaboration between teams from various enterprises. When models are communicated there is a risk for error when potential problems or important dependencies between design objects are not detected. Control applications are often of a critical nature and there are strict requirements on performance, safety, and reliability.

Considering the work of control application engineers there are many different technologies and implementation platforms that the designer uses in the work tasks. For

this there is a lot of supporting information available to assist engineering such as information from related engineering, manuals and handbooks, specifications, and guidelines. This information is most often readily available in an electronic format but cannot be efficiently provided during design when it is needed. In addition, there is also tacit and collaborative knowledge, e.g. emails and notes from similar projects, utilization of which is difficult.

4.4.1 Categorization of Engineering Knowledge

Engineering of control systems involves using data and other supporting information from many sources but the utilization of the information content is often challenging due to heterogeneity and lack of structure. In [P4] a categorization of design information is presented to be used in engineering scenarios to promote integration and management of related knowledge. The categorization is utilized to combine knowledge in a number of OWL ontologies to unify supporting knowledge, and enable semantic integration in a timely manner during design. The categorization divides the engineering related ontologies into three different categories: domain knowledge, model instances, and use case specific knowledge. The first two categories have been briefly discussed in section 3.3.2 related to describing modeling language constructs as an ontology, and model instance objects as ontology individuals, respectively.

In addition to information extracted from the modeling language, the domain knowledge may include information rooted in standards, manuals, specifications, and general practices used in the line of business. Also devices, components, and equipment described using ontologies can be classified as domain knowledge. Characteristic to this type of knowledge is the static nature of information and for the ontologies not to change frequently. From a modeling perspective the domain ontology describes the building blocks and the methods on how applications are designed.

Model instance knowledge represents the models under design or analysis. Concerning model instance knowledge of the transformations presented in section 3.3.2 the model instance ontology contains corresponding OWL individuals for the model element instances. For the OWL individuals the class types have been declared as well as associated data properties and object properties representing connections to other model objects.

Use case specific knowledge, on the other hand, refers to information that does not fall into the two previous categories. This knowledge is best described as information that is used in analysis and reasoning, e.g. rules on typical design issues, pattern-like

structures, company specific conventions, and concerns where special attention is desired. This information is often of a generic kind and it can be applied to concepts defined in domain knowledge ontologies, for instance. This requires concept and knowledge mapping so that information can be combined in a meaningful way. Also mappings of concepts between different domains or representations in different modeling languages fall into use case specific knowledge. Semantically enriched modeling concepts and tacit knowledge, for instance, can be stored and applied in various engineering scenarios to support design with equivalence mappings and pattern structures.

Figure 16 illustrates the relationship between knowledge according to this coarse categorization and presents the relationship between categories according to the previous definitions. The use of ontologies allows knowledge to be semantically integrated by linking concepts and inferring new information. It should be noted that currently only a small subset of information is described using ontologies. The increasing popularity of Internet of Things (IoT) [61] and Linked Data for industry [30], however, may change this in the future. Nevertheless, the method can be used to manage metadata of traditional resources and technologies to, for example, automatically extract this type of domain knowledge has been proposed [39].

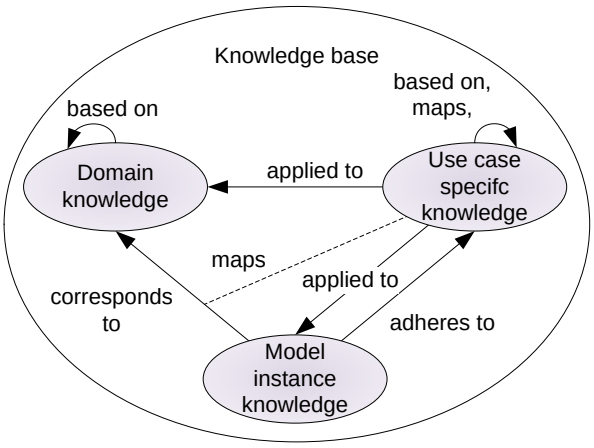

Figure 16 The knowledge in engineering applications scenarios can be divided into three main categories: domain knowledge, model instance knowledge and use case specific knowledge. [P4]

The categorization of information described in ontologies presents what kind of information is used in supporting engineering activities and proposes a classification guideline for organizing and managing related ontologies. The categorization, however, cannot be considered absolute as borderline cases can be identified where knowledge has characteristics from several of the categories. For example, design practices and

conventions can be considered to be domain knowledge on one hand, and, on the other hand, company or project specific practice and therefore classifiable as use case specific knowledge.

4.4.2 Semantic Identification of UML AP Model Objects

UML AP model reasoning based on OWL 2 DL ontologies was initially presented in [33] and extended to be used in engineering in [P4]. The OWL ontology representing the model element instances contains the details of the UML AP model including characteristics and structures such as Ports and Connectors as illustrated in Figure 17.

The UML AP models are structured and hierarchical in the sense that most of the elements can contain a number of nested elements in a long chain that ultimately on the lowest level define, for example, attributes based on which some elements are connected. Interpreting the model can be eased with simplification of structures for example by defining connections as graphs between the elements with rules:

```
AutomationFunctionPort(?prtA),
AutomationFunctionPort(?prtB),
Connector(?cnn),ConnectorEnd(?cnnendA),
ConnectorEnd(?cnnendB),
hasPart(?cnn,?cnnendA), hasPart(?cnn,?cnnendB),
hasLinkId(?cnnendA,?idA), hasLinkId(?cnnendB,?idB),
direction(?prtA,"out"^^string),id(?prtA,?idA),
id(?prtB,?idB),
DifferentFrom(?prtA,?prtB)->
hasPortConnectionOut(?prtA,?prtB)
```

From the example in Figure 17 the rule infers a hasPortConnectionOut objectproperty connecting the Y203 individual with the M100INT individual in the model instance ontology. Because of the fact that the target individual, i.e. M100INT, is an Interlock the port connection can be further inferred to be of type hasInterlockPortConnectionOut. Because of defined object property hierarchies also more general connections are automatically identified as illustrated in the figure. Connections of opposite direction (not depicted) are inferred as a result of declaring inverse properties in the domain ontology that defines the modeling language semantics.

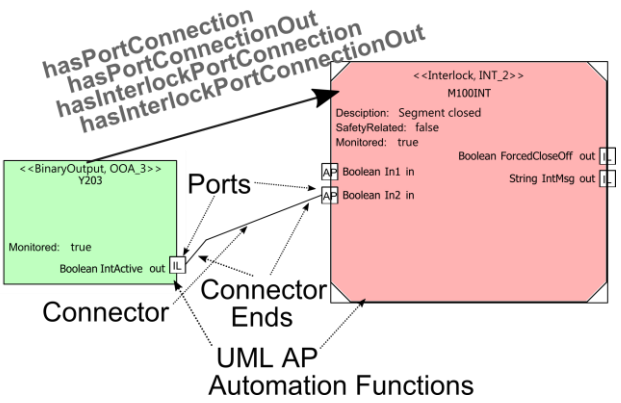

Figure 17 A connection is inferred from the port based connection between the two UML AP elements.

In order to enable information of a generic kind to be integrated some generalization of the concepts is required. For this a mapping ontology, which according to the categorization of section 4.4.1 falls into use case specific knowledge, can be used to align concepts. The model instance ontology contains individuals declared and identified as class types defined in the UML AP domain ontology. In the reasoning example of [P4] an ontology containing general engineering knowledge is used to express knowledge about common structures such as what a control loop, for example, is considered to be. In Figure 18 the element in the upper left corner is an AnalogMeasurement, defined by the UML AP domain ontology, and declared in the model instance ontology. Using the mapping ontology the AnalogMeasurement is generalized to a MeasurementComponent of the application model. The black bold markings in the figure indicate such mapping inferences for generalizing the OWL individuals in the semantic model.

Similarly also the port connection constructed earlier is generalized simply to a hasInterlockConnection (subclass of hasConnection) objectproperty directly between the element individuals. This is achieved with a mapping ontology defining a property chain that abstracts the UML AP specific port structure individuals and associated properties. The property chain identifies two different elements that have nested port elements connected via a port connection, and infers a simplified connection to be available instead.

With the aid of axioms directly connecting individuals further inferences can be made based on knowledge in the general engineering ontology. For instance, the Primary Controller and Secondary Controller inferences are a result of an OWL class definition that considers set points connected to another different Controller Component. This structure of chaining controllers is called cascade control and the purpose is to use a

regulated control value as the setpoint for a secondary controller to handle disturbances of a process, for example. Typically set point values come from AutomationFunctions such as Measurements and the control output value is transferred to an AutomationFunction of type Output. Based on the existence of cascade controllers within the ControlLoop the control loop can further be classified as a CascadeControlLoop. The automatic identification of this can be used to remind the designer of its existence and that special care should be taken in review or testing of those control loops, for instance.

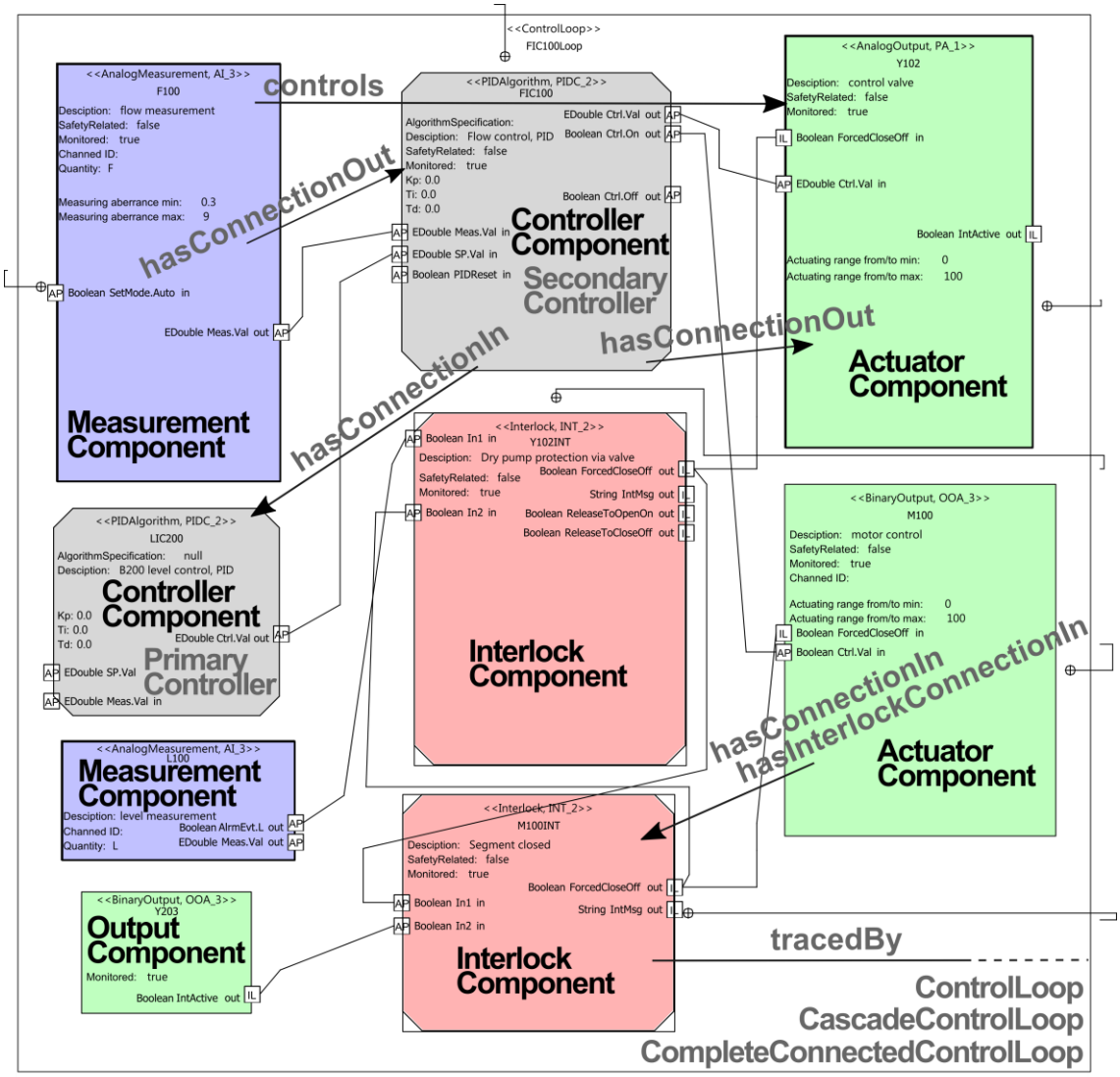

Figure 18 The semantic view of the UML AP model with generalized type inferences in dark bold marking and further inferences in grey bold markings. [P4]

4.4.3 Model Inference Using Knowledge in OWL Ontologies

Models described using Semantic Web technologies enable logic based reasoning to be performed on the models. OWL 2 DL used in [P4] is based on description logics. This means that the reasoning is monotonic, i.e. earlier conclusions cannot be invalidated based on the discovery of some other fact. As a practical implication, additional techniques are required if some fact needs to be altered based on some condition to keep the knowledge base consistent. OWL definitions are also not constraints for restricting the model content, and to be precise, they are used as axioms to infer knowledge and classify objects. This means that a too detailed and approximate ontology easily causes unintended classifications resulting in unexpected inferences.

An important concern for reasoning is the nature of Semantic Web and especially the Open World Assumption (OWA) [17]. OWA restricts deductions to those based on statements that are known to be true and, consequently, prohibits using negation as a failure in inferences. For example, ControlLoop(CL) can be defined as a disjointUnionOf CompleteControlLoop(CCL) and IncompleteControlLoop(ICL), i.e. declaring CCL and ICL disjoint and CL as the union of CCL and ICL. Given a set of rules on properties and relationships we can infer the instances belonging to CCL. Because of the OWA, however, ICL instances cannot be deduced as the remaining instances of CL not classified as CCL. In order to be able to make this assumption the remaining instances would have to be explicitly declared not to be of type CCL which cannot be done beforehand using OWL.

The control application model, however, can be seen as a closed space compared to, for example, how the Semantic Web is defined by [9]. While the OWA is justified for the nature of the Semantic Web [85] there are many tasks, e.g. the previous example, that would benefit from an approach where some of the inferences could be made based on a closed world assumption [52]. In order to overcome the limitations of OWA special techniques, e.g. [53], are required to close the world to allow such reasoning.

This can also be achieved using intermediary application logic such as program code. In [P4] this has been done using simple set operations related to the previously discussed incomplete control loop, for instance. Concerning the example the complete connected loops are identified by rules and class descriptions, and queried and removed from a temporary set containing all of the control loops. By using this type of programmatic closing for a known object set the remaining control loops can be deduced to be those of interest. It should be noted, however, that the facts are deduced outside the OWL realm and not added to the ontology. Axioms could be appended to the ontology by this same

application logic, if requested, but this would require additional mechanism to properly manage external knowledge inclusion from outside OWL to maintain the validity of the knowledge base.

It must be noted that reasoning on OWL DL ontologies is challenging concerning both time and space complexity. For large models the reasoning becomes computationally challenging even for simple appearing problems. As general methods to handle the complexity of large models the reasoning can be limited to a subset of the model at a time, if possible. For control application models this can mean reasoning on model elements one package at a time. This can be a problem when analysing model elements that have connections of importance to elements in different packages. Including linked elements from the first depth of other packages may not always be sufficient due to the structure or the role of the model element. Another approach is to divide the reasoning tasks to a number of smaller, although overlapping and redundant, reasoning tasks. For this method redundancy is often required e.g. for generalizations and inferring general knowledge that more complex rules are based on.

4.4.4 Reasoning for Model Analysis

There is information and knowledge that can be expressed as rules and structures of elements to describe constructs of interest, potential flaws and error-prone solutions. Considering UML AP, OCL [69] could be used to define constraints on models and to identify certain structures. OCL, however, is limited to the modeling language in question and does not support maintaining knowledge of a generic nature. For OCL expressing information of the previously mentioned kind would require binding it using modeling language specific details. This can potentially restrict using the information for other purposes such as modeling notations other than UML AP.

In [P4] linking of generic engineering knowledge to control application models is proposed using OWL ontologies. To apply this generic knowledge to a model being designed a mapping ontology is first applied to the model ontology to generalize the concepts as presented in 4.4.2. This allows the generalizations of the model elements to be used in making further inferences when identifying the generic and larger compositions of interest. The relationship of the ontologies is illustrated in Figure 19. Because UML AP is based on UML and SysML also their corresponding ontologies are present and available to be used for reasoning purposes.

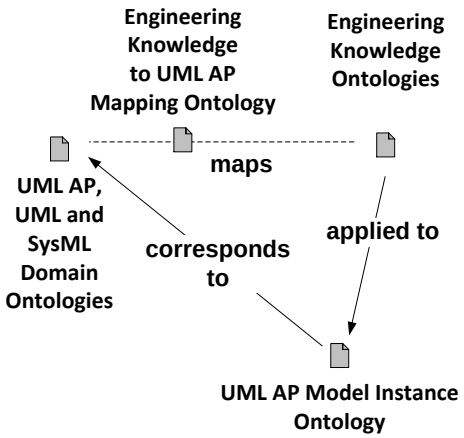

Figure 19 A mapping ontology is used to align UML AP constructs with generic engineering concepts in order to apply engineering knowledge of a generic kind to UML AP model instances.

To improve the process of reviewing designs, for instance, there are many potential use cases where knowledge of a generic kind can be applied to models being designed. An example of typical tasks is checking that certain criteria are met e.g. in property parameters for control application blocks. Similarly the structural correctness of connections between some of the elements can be determined in evident cases, i.e. checking erroneous connections such as an interlock outputting to a control algorithm set point, or assuring that a control loop is complete. A complete control loop is identified based on the inclusion of a measurement or a setting that is used to either directly or indirectly using a control algorithm to control an output, e.g. an actuating functionality. The control loop can be further classified as completely connected if the minimum amount of proper connections between certain elements is defined. Using this type of approach some of the typical mistakes in design can be automatically identified.

When developing systems with safety requirements it is often the case that interlocks are implemented using separate measurements than those used for regulatory control. This allows, for example, using fail safe components for safety features which should be followed if such kind of a separation is intended. Using a simple class definition the general kind of MeasurementComponent objects that are used both for interlocks and regulatory control can be identified as InterlockSharedMeasurement:

```
InterlockComponent and hasConnectionIn some
(MeasurementComponent and hasConnectionOut some
ControlAlgorithm)
```

To enhance the quality of the design process, traces can be used to verify design between model elements. In UML AP trace relations are used to trace Requirements to

AutomationFunctions. For example when reviewing the control application the traced Requirements can be queried programmatically for a particular part of the model for easier checking of the designs solutions. In a similar manner it might be beneficial to also query those Requirements that have not been traced to designs either directly or by Requirements that the initial Requirement might have extended.

For this purpose a model analysis Web Service was developed in [P4]. The service operates on OWL ontology semantics, and utilizes the Pellet [102] reasoner in the background. The Web Service exposes three methods: GetDefaultOntologies, KbIsConsistent, and AnalyzeModel. The first method returns a list of preloaded ontologies known by the service that can be utilized, and the second method performs a sanity check that the knowledgebase is consistent. The AnalyzeModel method is more complex as it is designed to be generic and utilized in various scenarios. The method takes an AnalysisQuery message as input that includes the ontology to be analysed as well as a list including URLs to other ontologies to be also included in the reasoning. The query part consists of two types of queries: objects to identify and complement objects to be identified. The first part includes a sequence of objects of interest to which corresponding lists of individuals is returned in the response. The second query type consists of a base class definition and a sequence of class definitions to be excluded when returning a list of individuals that are a complement set to those excluded.

The model analysis service has been designed to be flexible and it does not force the use of any specific ontology unless declared in the request. The model ontology can be implemented according to the approach of separating modeling language constructs from instance data, as presented in section 3.3.2, and transformed from UML AP using, for example, a XSL transformation such as the one presented in section 3.3.3. The service, however, can be used to analyse also other kinds of models within the limits of the analysis features described earlier. Because the knowledge base is built for each query the user can specify on what knowledge the reasoning should be based on. The analysis service is utilized in the next section and also later on in chapter 5.

4.4.5 Concurrent Engineering Support Using Ontologies

The model analysis service is such that it can be utilized in model checking tasks that are not dependent on the immediate response, i.e. the checks can be performed periodically for example when new revisions are reviewed. Delivering design-time semantic analysis, on the other hand, is less trivial when time and space constraints are present. In order for the additional semantic features to be useful they must be delivered

timely. Semantic awareness can be achieved efficiently using the transformation approach presented in section 3.3.4 that simultaneously maintains an OWL knowledge base of the model being designed.

Concerning reasoning it is not always practical to do all reasoning tasks as on-line processing every time the model is changed [P4]. Many of the reasoning tasks presented, and especially those related to model checking discussed in the previous section, are computationally complex and time consuming when the model being analysed grows to hundreds of objects. Therefore it is suggested that concurrent model reasoning is limited only to a subset of the model at a time, or limited to occasions when processing time can be allocated for more extensive analysis. This, however, again depends on the structure of the model, i.e. on how elements in sub models are connected.

The semantic knowledge base presented in section 3.3.4 also provides refined subscriptions to events of interests occurring in the UML AP model editor. Events that trigger execution of more complex analysis can be, for example, when the model is being saved and when the point of focus in the model is changed, e.g. diagram views are switched. These are also the events utilized in the prototype IDE integration of the model analysis Web Service presented in the previous section.

Figure 20 illustrates the setting for which an AnalysisView plug-in has been developed to integrate the semantic knowledge base with the functionality offered by the model analysis service, and present the results to the user. To perform analysis on the model the AnalysisView component extracts a serialized OWL ontology from the semantic knowledge base, and provides it as input along with references to additional ontologies. Naturally, the input also includes the queries for objects and structures that are of interest. The analysis service returns an XML message response that includes the references to those individuals matched in the different analysis tasks. The analysis view uses these references to list the individuals according to the queried classes, and presents them in a user readable format with names identifying the UML AP constructs. Based on this information the designer can review the constructs and parts of the model that based on existing engineering knowledge are considered erroneous or known to be prone to error.

In the prototype, the analysis view is implemented as a separate textual Eclipse view. From a usability point of view the method could be improved to highlight model elements directly in the UML AP model editor. Furthermore, implementing such features would require additional modifications to the UML AP editor as well.

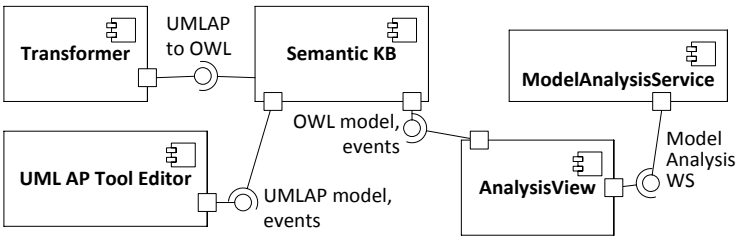

Figure 20 The architecture of utilizing the semantic model analysis service in an integrated development environment.

Reasoning tasks such as generalizations and simplifications of structures can be executed concurrently as only a moderate number of axioms at a time are added to or removed from the knowledge base. This type of reasoning for the model element that has been selected, the structure that the element is part of, or the control algorithm in question, for example, are good candidates to which other existing information can be integrated. This information of a supporting or otherwise related kind serves the purpose of assisting the control application designer.

The semantic representation of the model allows contextual information to be inferred and used as a hint to automatically provide suitable information related to the task at hand. For example such information that is described and shared similarly to the Linked Data initiative is viable for integration [34]. Linked Data is interesting especially because its philosophy is based on the idea that resources are described using machine interpretable semantics, and that these resources can be shared and linked globally. For semantic descriptions this could at the very least provide interoperability of common concepts e.g. for describing domain knowledge.

To explore the idea the AUTAKI concept of work support for control engineers [96] was conceptualized further to utilize descriptions based on Semantic Web technologies. Figure 21 presents the implemented prototype architecture in which a Work Support View was developed to automatically show information related to the engineering task at hand. The semantic knowledge base was used to retrieve the engineering context, and generalizations were constantly being inferred in order for the mappings to link material that was deduced to be related. For the prototype a new OWL ontology was constructed that provides URLs to Web content representing the related material that could be presented inside a browser component of the IDE. As an example the ontology was developed to contain links to general information such as different control algorithms and cascade control structures, but also to some type circuits of an implementation platform, i.e. to represent data that is typically available in manuals or datasheets. In a similar fashion the information of the Annotation Service in AUTAKI [32] could be

integrated as well if the service was made semantically aware of the types of objects to which comments and discussions are accumulated.

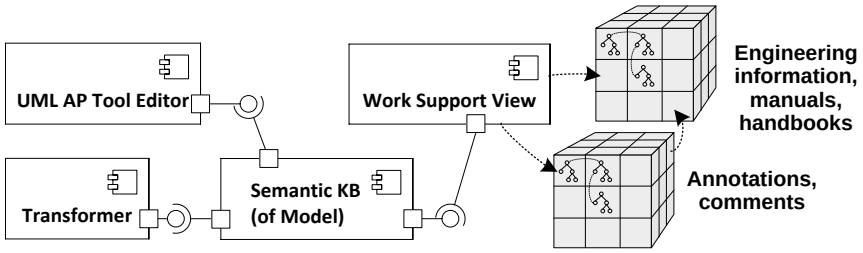

Figure 21 The semantic knowledge base is used for concurrent work support to automatically include relevant supporting material. For the prototype an OWL ontology was constructed with links to additional supporting material that can be displayed in a browser view of the IDE.

4.5 Discussion

The UML AP based model-driven development method has been designed with existing practices and conventions of the domain in mind. To evaluate adaptation capabilities to preceding engineering, different types of typical engineering data using various formats have been tested by implementing several import tools. The requirements model has shown to be sufficient to capture this data in a structured and unified way. Following this, a development chain has been designed promoting domain-specific concepts, platform independent design, reuse of solutions, and support for existing well tried implementation platforms.

One of the primary goals for the AUKOTON development approach is to make control application engineering more efficient and increase the quality of designs. For example for the requirements modeling phase it can be argued that the number of faults due to manual input and processing of data is reduced due to automated import functionality. The method is also scalable as models can be constructed into packages according to subsystems, for example, to ease management. In principle, separate UML AP models, i.e. model files, could even be integrated to form a larger application model but this has not been studied nor has any modeling support for this been developed. At a minimum this would require handling the connecting elements between different model files.

It could be argued that formalizing requirements modeling for control application development is merely shifting design to an earlier phase of development. In engineering of industrial control applications this is, however, desirable and beneficial. Background information, e.g. diagrams, schemes, listings and process models, already

are of a structured nature and represent information that impose classifiable requirements on the application software.

Functional modeling represents logically a phase in the development that allows domain constructs to be used in the application development without restricting possible implementation platforms such as PLCs, proprietary DCS platforms of different vendors, or any other implementation technology. Declaring functional, platform independent modeling as a distinct design phase enables, in addition to solution reuse, better organization of design work. This includes interfaces and interaction points between designs of different subsystems as well as work phases between different teams or enterprises. This is also supported by keeping models of different phases separate, instead of constantly elaborating a single model instance, to retain the source information intact and maintain backward traceability of design decisions.

The AUKOTON development approach can be adapted to typical DCS and PLC platforms used in the domain. This is especially beneficial as the established platforms are typically well tried and have many valuable characteristics as previously discussed in chapter 3. The transition from functional models to platform specific models is considered the third phase of development which can be delegated in a distributed engineering setting.

A known challenge of model-driven and model-based methods is compatibility of models between iterations. For example, if a model initially belonging to a previous design phase is updated at a later stage of development the implications to elaborated models are unknown from an automated processing point of view. This has been analysed by [36], for example. This is also an acknowledged drawback of the AUKOTON approach and backward transformation of models is close to non-existent, e.g. transforming functional models back to requirements. A trace based approach for supporting round-trip engineering and evaluation of some of the quality attributes has been proposed for embedded systems development [20]. In general, however, round-trip engineering is challenging even if every single action performed would be separately traced because usually there are no guarantees on what grounds the previous decisions have been made if some details of the context are changed.

Classification of model elements using OWL based semantics can improve the understanding and enhance the meaning of concepts used in development compared to what the modeling language definition provides. The semantic descriptions allows new meaning to be inferred for instance of structures that are out of scope for the metamodel (section 4.4.2) and to combine objects under design with other existing knowledge

(section 4.4.5). Prototype demonstrations have shown that OWL based semantic descriptions can be used concurrently with traditional modeling methods to enhance engineering tasks and provide additional information via semantic linking of resources.

The use of ontologies requires specification of ontologies, e.g. in the case of general domain knowledge, but also management of this information. First, the semantic descriptions have to be found and adapted to its use case. Secondly, the resulting knowledge base as a whole needs to be maintained. The knowledge management aspect is emphasized when linking data and resources from external sources which have their own interest to maintain and develop new information.

Not all related information and supporting material is described using ontologies, and this will probably remain the case in the nearest future. Indexed and semantically annotated material is needed to be able to provide supporting material in a context aware manner. Nevertheless, ontologies can provide semantic links to external material not contained in ontologies, and be used to provide coarse-grained metadata for resources such as HTML pages, books, specifications, manuals, and user guides.

OWL was chosen as the Semantic Web description technology mainly due to its reasoning capabilities. It is acknowledged that RDF could be used in many cases to achieve similar results. Especially only for linking resources and querying of objects this could be sufficient. The technologies are, nonetheless, sufficiently compatible and RDF based resources such as external supporting material can be linked to OWL based descriptions e.g. to support engineering tasks.

One might ask why the model-driven method does not include Semantic Web based technologies directly in the models. The main reasons are to keep the models transferable and compatible, and not too complex to put strains on processing. In the current stage of development the OWL based semantics are more purposefully delivered as an adapter based view to the model content. Compared to approaches that integrate ontologies with some other modeling language the approach proposed in this thesis uses this knowledge primarily to make the engineering environment more intelligent. This way the ontology semantics can be seen as an additional modeling layer on top of the metamodel based modeling in the case of UML AP or some other modeling language. This is then used to support engineering tasks e.g. using intelligent features of the IDE.

UML AP and similar metamodel based modeling languages provide syntactical conformance of the models while OWL brings additional semantics to the models under design. What OWL lacks in restrictions and regulations is compensated by the modeling language, i.e. the metamodel restrictions in the case of UML AP. This, however, does

not ensure similar validity of the semantic descriptions or the resulting knowledge base when other information is integrated and unintended semantics may occur.

OWL based reasoning easily becomes computationally challenging even for simple appearing problems. The computational complexity can be eased locally by planning how and when to do reasoning, and stating information more explicitly. However, the fundamental challenge remains as the nature of Semantic Web relies on integrating and linking externally defined knowledge, i.e. resources out of control and managed by somebody else.

5 Services and Process Management of the Engineering Lifecycle

Chapter 3 discussed the information content in control application models from the point of view of domain requirements and distributed development. Chapter 4 presented engineering methods in control application development. This included a model-driven engineering approach addressing the requirements as well as the use of additional semantic descriptions to support engineering.

In this chapter the engineering processes are discussed from the point of view of transforming engineering activities to repeatable and well-defined services supported by information systems. The aim is to promote the use of such engineering services and to facilitate related information exchange. Business process modeling methods are then proposed to ease and improve process management of control application engineering, and ultimately also to automate execution of some of these processes and their engineering tasks.

In addition to engineering during design of a manufacturing plant the resulting information model and the engineering data is of relevance also during the later plant lifecycle. In the lifecycle the design phase is relatively short compared to the expected life span of the plant during which operation and maintenance are of high importance. O&M also includes engineering tasks and for these purposes convenient and up-to-date access to existing plant information models and data is required.

Cloud Manufacturing is an interesting concept with analogies to the engineering services approach discussed in this thesis. On one hand, Cloud Manufacturing adopts service technologies for integration and cloud computing technologies for scalability and dynamic reallocation of resources. On the other hand, Cloud Manufacturing transforms the actual manufacturing into utilization of virtualized manufacturing resources through composition of virtual services made globally available. [130] Research in this area has been conducted related to, for example, service descriptions of offered manufacturing services [93], manufacturing operations transactions [131], and dynamic operations scheduling and service composition [54][132].

The use of an Automation Service Bus (ASB) to integrate processes, models and tools for quality and process improvements in engineering has been discussed in [10] and [63]. Concerning engineering it is stated that an ASB can bring significant benefits to a heterogeneous engineering environment by awareness of changes in the project environment, up-to-date documentation, data collection and analysis, and quality

assurance. The use of semantics has been further considered [64] and a general engineering knowledge base is proposed for integrating and mapping engineering data between tools used by different parties. A service based approach for developing embedded systems models in a distributed environment using Semantic Web descriptions has also been proposed [112].

E-maintenance is a term used to describe software platforms for integrating maintenance related data using Web based technologies and middleware solutions [3]. The use of these platforms to integrate business processes has also been proposed [35] but sufficient means have not yet been developed or implementations are considered a task for enterprise systems [47]. The information systems in use, including field devices, are not all capable of SOA based integration as many legacy systems are still in use. Evolving systems towards SOA requires detailed business process analysis as well as architectural analysis [41].

Engineering enterprises are under constant pressure to improve efficiency and quality of their work. Another important business driver from the point of view of this thesis is the increase in cooperation and the way large engineering projects are conducted as collaborative efforts of interweaved engineering activities[7]. The first point calls for new methods and tools for how work is executed, i.e. improved engineering methodologies and means that support or even automate some of the tasks (section 5.1). The second point is related to how engineering activities are organized and defined to be used in a collaborative and distributed setting. The latter also implies management of engineering processes which will be discussed in section 5.2. The manufacturing and production industry has to deal with similar issues while operating the plant and means to improve this are presented in section 5.3.

5.1 Service Enabled Design and Engineering

5.1.1 Engineering Tasks as Services

From a business perspective a service can be defined as the “application of specialized competences (knowledge and skills) through deeds, processes, and performances for the benefit of another entity or the entity itself” [115]. A key characteristic of services is to

[7] Collaboration is often needed due to more advanced features being used that broaden requirements on expertise from various fields of engineering.

create value. In service science services can be seen as systems in which one entity applies competence and another integrates this competence, but also determines the perceived value [104].

Engineering tasks can be organized according to this service definition and be utilized in a larger context such as a virtual enterprise[8]. A general rationale for establishing virtual enterprises is to be more capable as a group sharing and combining specialized expertise, knowledge and resources, and to be better able to respond to various needs. This structure could bring agility and flexibility to engineering processes, and enable faster adaptation to changing needs. This also enables companies to focus on core competence areas, and utilize specialized expert services, e.g. advanced control design, simulation, testing, verification, and validation services, when needed.

In addition to the aforementioned services also the engineering tasks of different engineering disciplines as a whole can be considered as services. Service candidates can also be identified by analysing interactions in engineering processes between engineers, teams, and enterprises. These can then be divided into sub sections, and in turn be delegated to a new group of services. This is encouraged from the point of view of managing both information exchanges as well as engineering work in large projects where tasks can be assigned to different engineering companies, i.e. service providers.

It has been envisioned that engineering services will be requested in the future similar to how weather services and stock quote data is retrieved today [8]. [P5] conceptualizes the use of services as building blocks for engineering activities, and using these services to compose engineering workflows (see Figure 22). It is common for large engineering projects to be distributed among a number of enterprises. It is also often the case that these enterprises utilize subcontractors to perform some of the tasks or to design subsystems, for example. Many of these tasks are interdependent hence requiring task integration and engineering data exchange. This is also the case for projects where control application engineering is a part of a larger engineering undertaking. Control application engineering may thus require information from e.g. process design and instrumentation or data related to existing and previously engineered systems.

From an information systems and SOA perspective OASIS[9] defines a service as “a mechanism to enable access to one or more capabilities, where the access is provided

[8] Ad-hoc consortium of enterprises or experts that in cooperation share skills and resources, and often rely on telecommunication and information networks.

[9] Organization for the Advancement of Structured Information Standards

using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description" [67]. If the engineering services are defined repeatable, including specified inputs and outputs, the organized activities could be utilized as service offerings implemented and supported by information systems.

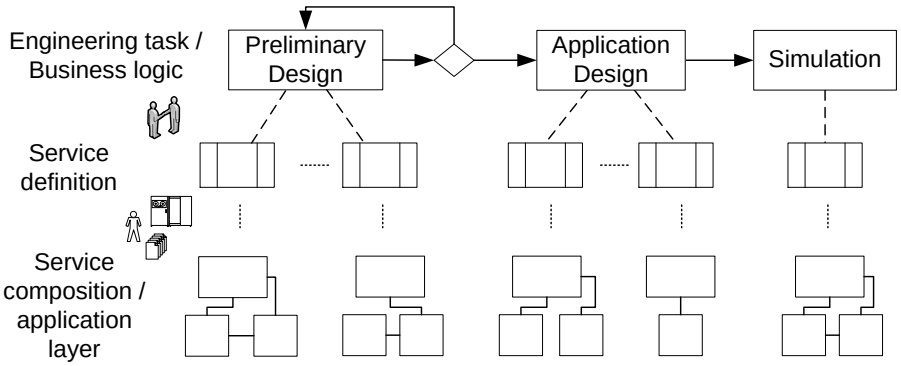

Figure 22 Services, whether manual or automatic, can be supported and implemented using information systems. Services can then be composed to engineering workflows if well-defined service definitions that standardize the information exchange are specified. [P5]

Many services that perform simple actions such as model checking or automated simulations can be implemented fully automatic using information systems. Naturally, there are, and will still be, many engineering services that cannot be automated as genuine engineering work is required. Nevertheless, manually performed services can be provided and consumed as services implemented using information systems and exposed e.g. as Web service interfaces. Shifting design to using services of this kind would have an impact on how engineering work is managed. It will also force engineering information exchange to be standardized unambiguously for services to be able to define what information input is required and what the expected output is.

5.1.2 Requirements for Using Services and Shared Model Information

In general, information systems and computer infrastructure are increasingly used as services and virtualized to be run on data centers and in cloud computing environments. When services and computing resources are shared and utilized more widely there are several positive effects achievable such as improved efficiency and performance, and lower costs of operation. In addition, this can also increase dependability, reliability and fault tolerance of systems developed in cloud environments when resources can be dynamically allocated at run time. [16][58]

A shared service framework, as proposed in [P5], would be of common interest if services and engineering data could be utilized and accessed in a standardized manner.

This would enable engineering offerings to be used through services that integrate information exchange between different engineering systems and tools. These services that are supported by information systems can be exposed as invokable service interfaces. Some considerations should be taken into account when planning to use this type of services as building blocks in compositions of engineering activities.

First of all, the service interface description needs to define the service, i.e. the action it performs. The service description includes, at a minimum, a service interface description that details the input data as well as the output data in relation to its offered operations. The input definition needs to declare all the data that is required in performing the offered service. Similarly, the output definition defines the expected output, but may also specify the output after error or unsuccessful completion.

The information content exchanged while using the service should be based on acknowledged formats. The data passed to a service or returned as a result should be based on standards or commonly agreed practices to facilitate provision of information as well as utilization of existing data. The information content needs to be structural and in a computer processable format. It is also beneficial if the data is of a fine-grained nature, i.e. divided and structured in sufficiently small and distinguishable fragments. This aids programmatically accessing and modifying the data in transformations, for example.

Concerning distributed engineering environments the access to up-to-date information is emphasized when shared resources are utilized, and data is produced and consumed by different parties. This is an important issue in settings with multiple participating companies in which information should be shared only with those actually needing it because of immaterial rights but also due to security and safety reasons. Access to relevant previous engineering data, however, is required in different engineering tasks. Control application engineering, for example, may require process design data. Naturally, this data can be provided explicitly per request, but this would limit utilization of automatic processing features. It would prevent development of, for example, automatic functionalities monitoring the initial source information for unexpected changes during later stages of development. The aim is, nonetheless, to share resources to improve the efficiency and quality of engineering in its entirety of control application development and related disciplines.

A service based operation model implies more than information system implementation. The use of information systems supported services also has an impact on how engineering is performed including subdivision of project responsibilities and legal

commitments. It is not uncommon for engineering to provide the minimum sufficient amount of engineering data to comply with project deadlines and later on provide an updated version. The initial versions can include errors or otherwise require manual resolving. It is noted that issues of this kind need more detailed service level agreements if improvements to information exchange is to be pursued in the proposed manner.

5.1.3 Considerations for Enabling Technology

The service architecture concept outlined in [P5] is illustrated in Figure 23. The figure represents engineering activities distributed into service units for different phases of development. From a technical perspective a service-oriented architecture is proposed that in combination with standard Web technologies provides means to create unified access to distributed services and resources. Essential to the concept is a logical integrating layer for managing the use of both engineering resources as services as well as the engineering data. For information exchange this would enhance sharing of mutual information instead of relying on common point to point data transfers.

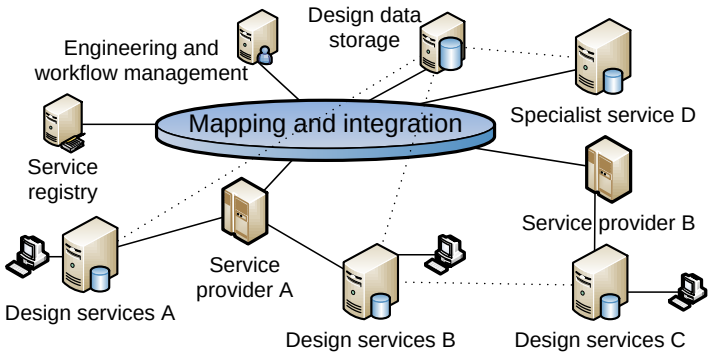

Figure 23 The concept of a shared service framework integrating engineering services and information exchange in an engineering setting of multiple participants. [P5]

The services can be implemented, for example, using common Web service technologies such as SOAP or REST style service interfaces. Information exchange can be implemented using XML based descriptions that enable platform independent data transfer, but also the use of XSD schemas that can be used to specify standardized data formats. It is important to note that XML itself does not guarantee that information can be processed in a meaningful way. Standards, on the other hand, give context and semantic meaning to metadata concepts and the elements used to contain the actual data. If considered further, Semantic Web technology based descriptions can be used to enhance descriptions of control application models in a computer interpretable manner [P6].

Cases can be identified when simply invoking services and waiting for the result is not sufficient, and more advanced service utilization patterns are required. For example, services that are completed in time frames of days or weeks are asynchronous by nature, and modifications that e.g. affect multiple data sources can require transactional resource handling. Service operations may also fail due to various reasons and mechanisms that handle faults and errors are also needed. It can be concluded that another layer of technologies and logic is required for managing the use of these services.

5.1.4 Prototype Service Infrastructure and Exemplary Services

In [P6] a prototype infrastructure was developed to demonstrate the concept of using engineering services supported by information systems. The infrastructure was developed with flexibility and interoperability requirements in mind, i.e. to allow switching service providers rapidly in changing circumstances, for example. For this an infrastructure enabling dynamic binding of services was required including methods for providing supporting commodity services e.g. for transformations.

A service bus architecture, often referred to as an Enterprise Service Bus (ESB), was implemented for integrating services in a loosely coupled manner. The role of the service bus is to operate as a mediator of services by providing access to a registry of services, and functionality for transparency of service locations, protocols, interfaces as well as encoding of data [P6]. This includes routing service requests, responding to events, and invocation of functionality that is required in using the services, e.g. services for pre-processing or reformatting data content. Service bus implementations typically also include methods for controlling and monitoring routing between different information systems as well as features for governing the use of specific services.

The service bus that integrates information systems and data sources used in engineering, and illustrated in Figure 24, implements sets of input-output sequences for the mediated services. The bus connects information sources, applications, and engineering services with engineering processes. Depending on the needs the operational context can be defined as team, department, enterprise or even inter-enterprise wide collaboration configurations. Naturally, the degree of implementation depends heavily on the information system infrastructure readiness, and on how external service provider activities, for instance, can be defined and utilized.

In Figure 24 the plant model stands for the design repository used in the plant-wide engineering. The plant model provides the hierarchical structure of the facility under

design, and is used for linking and synthesizing objects and engineering of different disciplines. Knowledge bases represent information in various formats such as manuals and documentation but also structured and semantically described information (as suggested in 4.4.1). Transformation services are a subset of the commodity services and important for increasing interoperability of services and information exchanged in engineering. The service registry provides means for discovering and looking up suitable service providers. Human task management as well as project management represent typical project management tools used to assist in managing work processes. Engineering services, as discussed in the thesis, include the services exposed as invokable interfaces.

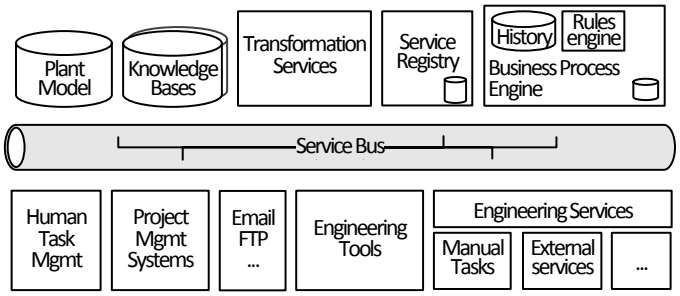

Figure 24 A prototype service bus infrastructure was developed to integrate services and data exchange in engineering of control applications. [P6]

The service bus was implemented using an ESB broker framework[10] for integrating the information systems. For managing the available services a service registry[11] was used. The system providing the plant model was based on a development version of the SEFRAM (Service Framework for Industry[12]) infrastructure for information exchange.

To demonstrate the infrastructure a number of services had to be implemented in addition to the integrated systems and their service interfaces. For example, to utilize the semantics based model analysis service presented in [P4], and discussed in section 4.4.4, a transformation service had to be implemented for models created in UML AP. For this the development environment independent model instance transformation approach, initially presented in [P3] and discussed in section 3.3.3, was utilized. The

[10] Mule ESB, URL: http://www.mulesoft.org/

[11] WSO2 Governance Registry, URL: http://wso2.com/products/governance-registry/

[12] THTH Association of Decentralized Information Management for Industry, URL: http://www.ththry.org

transformation was implemented on the service bus as an adapter service and a sequence was defined to route analysis requests of UML AP models via this adapter.

For testing purposes design services representing respective AUKOTON development approach phases were also implemented. Similarly, a service was defined for performing simulations manually. These engineering services mainly enhance digital information exchange as requests related to the actual engineering work have not been specified. The interfaces define the input data as well as links to auxiliary material that is of relevance in performing the work, and the expected output.

The approaches referenced in the beginning of chapter 5 are similar to the service bus proposed in [P6] but they focus mainly on the service interfaces or the models and do not address the need of a management layer for managing the engineering processes. The proposed service infrastructure will be utilized in upcoming sections where business process modeling is applied and the automation of these processes is considered. Also the use of these services in engineering processes is discussed.

5.2 Engineering Process Management

One reason to the lack of progress in tool integration for software engineering in general is that development efforts have been focusing too much on data instead of socio-economic dimensions [127]. Digitalization of engineering does not only imply using computer interpretable models, but also digitalization of the engineering processes [P6]. Therefore emphasis should be put on process management, including means to improve process management methods, and, if possible, also on automation of these processes.

The use of business process modeling methods to describe and improve management of engineering processes has been proposed in [P6]. In order to improve business process management in general the processes first of all need to be understood. This includes defining the tasks as well as the information content required and produced as a result. Based on this knowledge the processes can then be documented and formalized, e.g. using a business process modeling method. Special care is required when defining information exchange and integration of human tasks if information systems are to be used to support or even control the processes. The formalized processes include metadata information that can be used to monitor process activities and collect information. The gathered knowledge can then be used to improve the way of working by restructuring and reconfiguring the business processes.

5.2.1 Engineering Process Modeling Using BPMN

BPMN can be used to model engineering processes. BPMN 2.0 provides modeling constructs that are simple and intuitive and supports also management features for automatically controlling and monitoring processes being run. [P6]

BPMN can be used to model engineering processes on a general level as demonstrated in Figure 25. The figure presents engineering activities typical to design of a processing plant on a simplified and high level of abstraction. The individual tasks can be further detailed with more concrete process definitions as illustrated in Figure 26. In this way process model descriptions with varying levels of abstraction and detail can be developed for different purposes.

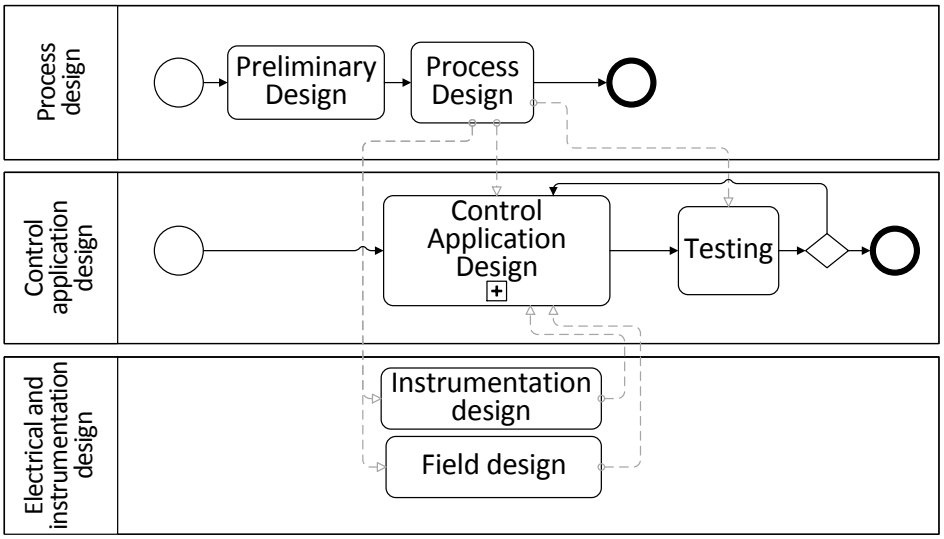

Figure 25 The BPMN business process modeling method can be used to model engineering activities on a general level without considering process details.

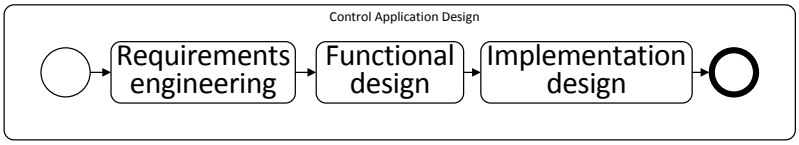

Figure 26 BPMN processes can be expanded with new process descriptions and the tasks can be detailed with more specific descriptions of activities involved in performing a task.

[P6] presents some examples of concrete engineering processes modelled as BPMN 2.0 business processes[13]. Figure 27 presents the example where a review process is initiated

[13] The BPMN process definitions were modeled using the Activiti Designer that is part of the Activiti BPM Platform, URL: http://www.activiti.org/

by defining the object to be reviewed, the group of users who the review is assigned to, and the number of reviews that are required for the review to be completed. The reviews are being performed manually as human tasks and after sufficient number of reviews is completed the result is notified to the initiator, and the model status is changed to having been reviewed.

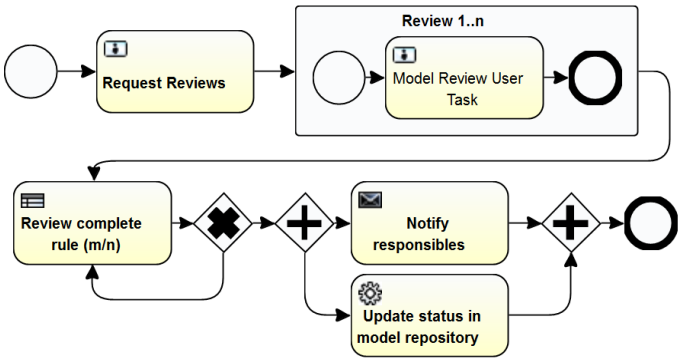

Figure 27 A review process of examining a model, sending notifications[14] of the results and updating its status in the project repository.

A similar example is the business process of performing simulations as a part of the control application development process as depicted in Figure 28. In the process the simulation task is first specified before being assigned to a person responsible for performing the simulation, e.g. a simulation specialist or an external company offering simulation services. In the simulation sub process the expert retrieves the model in question, e.g. downloads the model using his tools, and makes any necessary preparations, performs the simulation and finally reports the results. The results can then be evaluated and, if necessary, new engineering tasks initiated for correcting any defects found in the simulations.

Business process modeling can be used to describe the engineering processes, and standardize the way of working. Composing engineering processes from smaller sub processes as well as service tasks can also have a positive impact on engineering process management. BPMN diagrams accompanied by textual descriptions have been shown efficient in communicating business process descriptions [81]. This makes way for communicating and further improving the processes, and finally increasing efficiency of engineering and improving the quality of systems designed.

[14] Activiti Designer implements the non-official BPMN mail task as a dedicated service task.

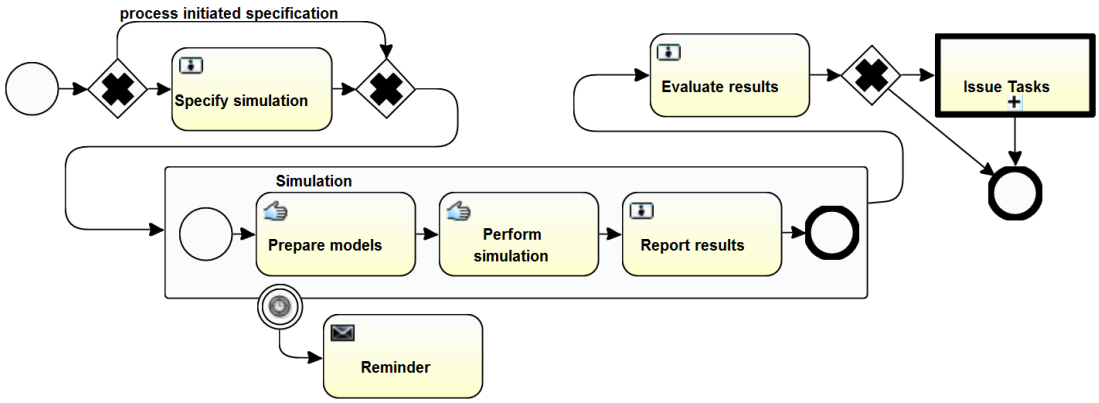

Figure 28 Simulation of control application models can increase the quality of designs if models are evaluated during design.

5.2.2 Executable Processes and Task Automation

Since the introduction of business process execution semantics in BPMN 2.0 the business process models can be run in a compatible engine. Figure 25 and Figure 26 presented engineering processes on an abstract level. The model review process of Figure 27, on the other hand, is more concrete in its tasks, the information required and results of its execution. These include engineering information as well as metadata information from process execution worth collecting automatically.

Using a business process engine the execution of the model review process can be automatically documented along with the results of the reviews. After the reviews have completed the process automatically executes the two final parallel tasks. The first task concatenates the reviews and sends a notification to the responsible. The service task, on the other hand, invokes the project model repository and updates the status of the object as reviewed. This requires implementation of user interfaces, i.e. user input forms, in engineering or project management tools to integrate the human tasks, and programmatic access to the project model repository.

In [P6] the business process engine[15] is integrated to the service bus presented in section 5.1.4. The execution engine allows exposing process definitions as invokable services that can be defined to accept parameters known by the process. The process depicted in Figure 28, for example, can be implemented in this manner. The process has an alternative workflow in the beginning that allows the process to be programmatically invoked and omit the first human task of specifying the simulation. This enables

[15] Activiti Engine from the Activiti BPM Platform, URL: http://www.activiti.org/

specifying simulation procedures directly from other processes or by tools that are integrated to the engineering service bus, for example.

The business process model of Figure 29 demonstrates how the semantic model analysis service (see section 4.4.4) is used with UML AP models. The process definition can be initiated for execution either manually or programmatically by specifying the analysis parameters, i.e. the model to analyse and the knowledge on which the analysis is being performed. However, for this process the service bus infrastructure dynamically looks up an available service from the service registry and binds the execution to that service endpoint. Because the model analysis service operates using ontology concepts the bus utilizes a mapping service, as explained in section 5.1.4, to transform the UML AP model to a corresponding OWL representation before calling the service.

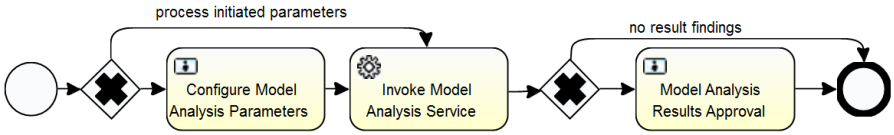

Figure 29 Automated model analysis of UML AP models is based on invoking the semantic model analysis service. With the support of the service bus infrastructure the model content is first transformed to an ontology presentation before calling the service bus mediated analysis service.

The process of Figure 29 was implemented also to demonstrate monitoring events of interest and to trigger execution of processes, e.g. for reviewing and analysis. In engineering processes such events are, for example, updates and changes to models and source data. In [P6] a monitoring service was developed as a part of the engineering service bus to periodically observe the plant information model for changes in objects. Several instances of this monitoring service can be run and configured to observe different parts of the plant model by configuring the search parameters accepted by the search API of the SEFRAM plant model search interface. The results are returned via the service bus that maps the responses to the desired action, e.g. detection of a new UML AP model file initiates starting a model analysis process. The reuse of these process definitions is also illustrated for the generalized AUKOTON development approach in Figure 30.

Service bus architecture supports the use of business processes to define workflows and utilize the services in engineering. To provide flexibility and ease the utilization of individual services in process definitions the service bus offers mediated access to services including commodity services. The developed prototype infrastructure supports definition of subscriptions for events of interest and thus also event-driven process execution.

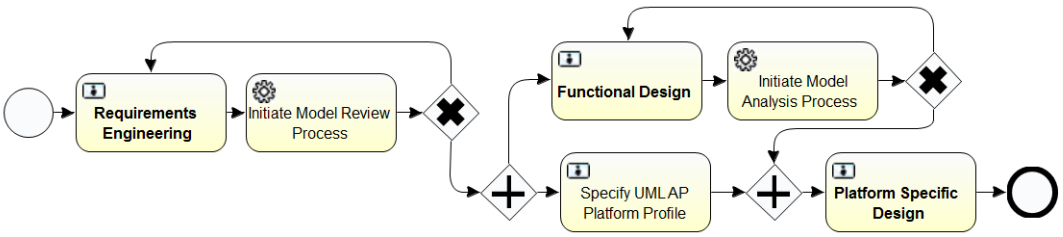

Figure 30 The generalized AUKOTON development approach including invocation of the predefined model review process and the model analysis process.

5.3 Models and Services for Operation and Maintenance

The plant lifecycle includes preliminary design, engineering, construction, operation, maintenance, and finally disposal and recycling of the plant. All of these phases generate information that is shared and utilized across the phases throughout the lifecycle. The operational phase of a manufacturing system or production plant includes operational activities such as process monitoring, manufacturing process improvements, and maintenance. Many of these activities include engineering tasks that utilize existing engineering information but also activities that update and produce new engineering data. It is also common to processing and manufacturing plants to have maintenance as well as engineering activities outsourced. Sufficient information exchange may not always be achieved as it can be challenging to implement in supporting information systems.

The operations model for manufacturing and processing plants has evolved in such a way that companies are focusing only on their production related core functions. This has resulted in outsourcing and provision of external service providers for supporting activities such as maintenance and engineering, e.g. when making modifications to the process. In addition to independent service providers also equipment suppliers have developed business models for providing services in addition to their product offerings [105]. Maintenance is typically outsourced for repetitive routine tasks or for specialist services that are not cost-efficient to have in-house [55]. For effective provision of these services, some of which are highly important for the overall efficiency, integrated asset management and sharing of mutually important up-to-date data is required in executing and coordinating work processes.

In [P7] business process modeling of O&M activities is discussed from the point of view of using services and SOA to implement supporting information systems. Accessing and using existing engineering data for these supporting applications is also considered.

5.3.1 Plant Information System Integration

Information systems as well as the data content used by different service providers are heterogeneous hindering the integration of systems and information exchange. This can lead to not accessing relevant information required or that some information has to be transferred manually. The plant model, for example, serves as an up-to-date logical model of the information related to the plant structure, the equipment, and the functions of the facility [P7]. This information is essential for targeting maintenance operations, and plant models can be used to link information of previously performed maintenance actions. For engineering it serves as a basis for performing engineering work in modernizations as well as for consolidating and merging the new designs.

The rise of features offered by modern DCS systems and intelligent field devices further increases the diversity of systems being used. For example to develop preventive maintenance schemes based on condition monitoring data various integrations to systems and devices are required. Access to this data is emphasized as service providers including OEMs gain more responsibility of the operations through partnering [2].

SOA is generally considered as a facilitator of loosely coupled information system integration, and it is also capable of cross-organizational information exchange. The architectural paradigm is well suited for the needs of O&M where autonomy of services of independent service providers is required, and the services are composed flexibly to match the needs of the business environment. [P7] SOA does not automatically solve the integration issues, but standardized and open access of systems is required to ease integration. The information content is also of relevance, as discussed in chapter 3, and standardized or otherwise approved information formats should be used to improve data interoperability and information exchange.

It is characteristic to service networks to change, e.g. on an annual basis, but more rapid inclusion of external specialist services may be requested as well. Considering the implementation this requires great flexibility in integrating services and data sources so that they can be accordingly adapted to changing needs of O&M business processes.

5.3.2 Utilization of Control Application Models During Operation

Control application models that are maintained e.g. as a part of the plant information model provide up-to-date descriptions of the functionality of the system in use. The application models serve as a basis for developing new functionality or improving the operation by introduction of new control algorithms, for example. Manufacturing as a process is also increasingly becoming more agile as facilities are designed so that they

can be reconfigured e.g. to change the manufacturing or processing methods, or to produce new product variations. If the existing equipment is to be used in the future also parts of the control applications can be reused after implementing respective changes and reconfigurations.

Another example using the control application models is simulation of how the application performs e.g. for different resources being processed. The concept of using simulation models in development of control applications was briefly discussed in [P5] in relation to simulation services. Generating simulation models corresponding to the control application is typically laborious. However, if control application models are specified using a standardized method there are chances of automatically generating parts of even full simulation models that represent the control application behaviour.

For UML AP this has been implemented for the functional models for which transformations to ModelicaML [97] have been implemented [116]. The original purpose of this was to enable simulating control applications in the early stages of development. Given the fact that a simulation model for the physical process existed, the two models could be integrated, and the functionality of the system under design evaluated. If, for example, the dynamics of the manufacturing process are changed due to later lifecycle reconfigurations the control application can be easily simulated. The suitability of the existing control application can then be evaluated for the updated process as well as to support the development of a reconfigured control application. The method has been further improved concerning simulation and development of safety related interlocks [120].

The application models can also be used for system integration purposes as the control models document the interfaces. If a new MES system is introduced, for example, the models can be used as a basis for designing system integrations. Concerning future self-configuring [110] and autonomic [113] systems the semantics of the models could be extracted for such purposes. This has been proposed for model-driven system development in pervasive computing applications using ontology semantics [103].

5.3.3 Business Process Driven O&M Application Development

Traditionally business process modeling has been viewed as a method for documenting and organizing business processes. Business process modeling can also be used to guide and steer development of software applications and information systems that support O&M [P7]. Software used to support O&M activities need to follow the principles of

the O&M business processes, i.e. distributed operation, lean management, and flexibility of the business environment.

Using business processes to model O&M activities has many benefits. The process models help to understand and organize the way of working. When tasks, involved participants, and information flows have been identified the process can be studied and further improved. The modeling can also assist in communicating processes to involved participants to clarify responsibilities and to describe the desired workflow.

Improving processes often means reducing human labour and implementing automated functionality in information systems. Implementing O&M processes executable allows automation of manual routine tasks. Information system implemented processes can also be executed periodically or initiated as a result of an event. Executable processes also enable automatically documenting and storing the individual executions, e.g. for future reference or analysis. This can provide valuable information of frequent malfunctions or error-prone devices, for example.

A compelling method is often to connect auxiliary systems directly to the systems that are used at the plant. These types of integrations, however, are typically point-to-point and have to be customized for each connection. When service architecture is the foundation for integrating information systems the orchestration can be implemented using more advanced methods that also enable easier reconfiguration. For example, using a service bus architecture, as presented in section 5.1.4, the services can be bound dynamically for each request. For O&M this translates to selecting service providers from a registry to which maintenance tasks can be delegated.

There are a number of different methods for implementing executable processes based on information system services. Many proprietary solutions offer tools for declaring workflows in this manner, e.g. in ERP systems. WS-BPEL represents a standard solution that can be used to orchestrate SOAP Web services[16] and BPMN 2.0 can be used similarly as discussed in section 5.2. Ideally, the specification of engineering processes mitigates to composing services and business processes exposed as services.

By composing services based on business process definitions the applications can be developed to match the needs of real world maintenance processes, i.e. how work is actually executed. This, however, requires that processes are described accordingly and

[16] REST services can also be utilized in WS-BPEL e.g. by defining WSDL description for the interactions, and developing an adapter component for handling the REST communication.

that process definitions are also followed. Redefining a process e.g. for defining a maintenance task for a new service provider is straightforward given that suitable service interfaces exist.

In [P7] a condition monitoring process example is presented that can be run periodically to query the condition information of certain devices. The process is a composition of smaller processes that on the lowest level utilize service interfaces of the plant information model and DPWS [65] enabled devices (see Figure 31). The plant information model is the same development version of the SEFRAM framework used in section 5.1.4 and the following engineering process examples presented in section 5.2. The plant information model is used to query serial numbers of devices of interest, and the serial numbers are used in the discovery of DPWS device endpoints to query.

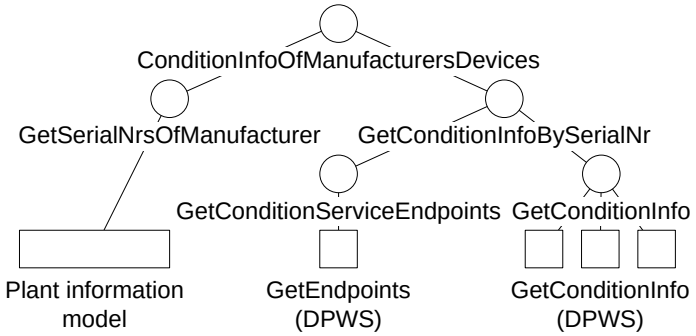

Figure 31 A maintenance process following up condition info of certain devices. The process is implemented as a composition of smaller processes that at the lowest level integrate service interfaces of a plant information model and DPWS enabled devices. [P7]

Figure 32 illustrates the service integrations for querying each of the devices and the automatic processing before reporting the results. The result of the process can also be used to automatically initiate a maintenance process e.g. for replacing the device. The process model[17] is based on BPMN 1.2 [70] and the executable process is implemented using WS-BPEL to which the model is transformed[18].

[17] The model is created with Intalio BPMS, URL: http://bpms.intalio.com/community

[18] Transformation to WS-BPEL is carried out using Intalio BPMS that supports partial transformation of BPMN models to executable WS-BPEL service orchestrations.

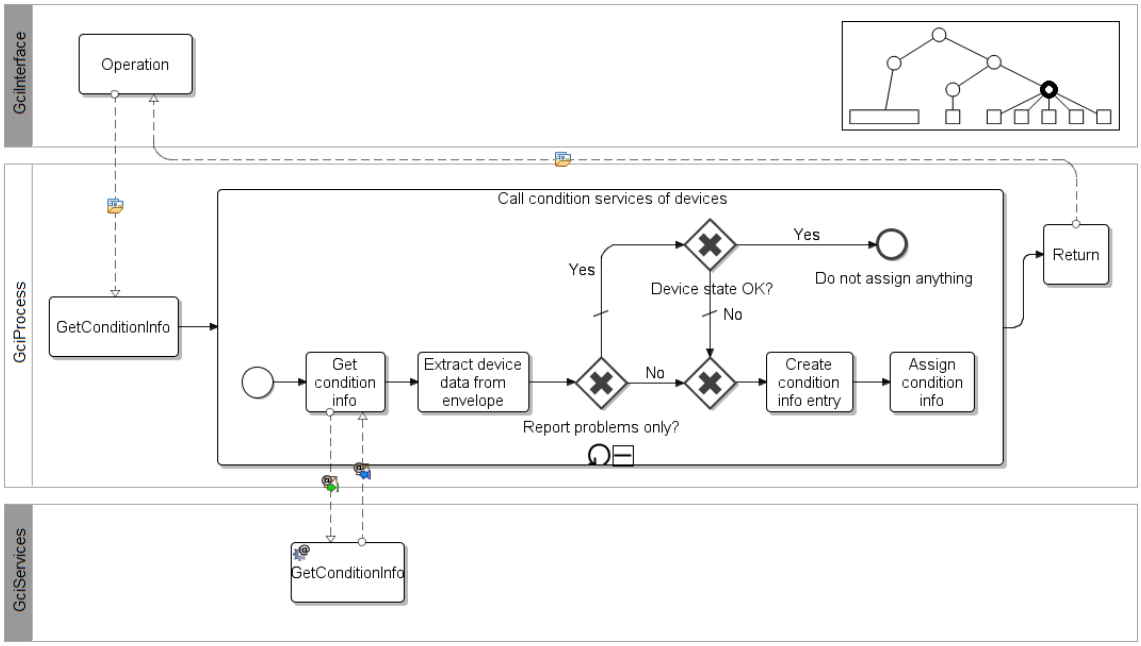

Figure 32 BPMN presentation of a sub process used to query and process device condition information. [P7]

5.4 Discussion

Organizing engineering work into service offerings within teams, companies, and even between companies can enable collaborative work contributions and expertise to be utilized more efficiently. Using flexible service compositions the service offerings can be combined in virtual enterprises to better cater the needs of large engineering projects. Utilizing services in daily operations as well as in various engineering related tasks throughout the plant lifecycle requires, however, a change of mindset. Outsourcing and the use of services should not be solely considered on short-term cost savings. It implies a transition towards a value network setting where expertise is combined from various sources in order to achieve a greater outcome in terms of lower costs and better quality.

Utilization of engineering service units improves managing engineering activities and enhances information exchange between participants. However, the engineering tasks exposed as services would need more specification concerning service requests and this would require further research focusing on the content of engineering, i.e. how engineering work is performed by individual designers. In addition, the collaboration scenarios also require detailed service level agreements in order to select suitable services at run time.

The information used in engineering of control applications and related disciplines is diverse and it is often required to use different modeling methods. Information content in various design systems is also diverse and for large projects the information amount is significant. Mapping and transformation functionalities are often needed to transfer information from one system to another. The service infrastructure therefore needs to support integration of applications and systems using several kinds of technologies. This includes the requirement to be able to create custom connectors for legacy systems. There are, however, apparent cases where information cannot be exchanged automatically e.g. due to restrictions in expressing power of different notations.

In order to provide flexibility and more efficient information integration the service infrastructure was developed with dynamic service selection using a registry and late binding of service components. In addition, transparency of protocols, service locations and information content was proven feasible using a service bus approach mediating the service requests. With flexibility and management of agile engineering processes comes also the challenge of how to efficiently implement secure access to data and services.

In related works, e.g. [10, 64, 112], the use of services in similar settings has been considered mainly from the point of view of service interfaces. Services, however, require another layer to be managed appropriately. Business process modeling of engineering activities proved to be a suitable method for defining engineering tasks of individual engineers, teams or companies. Process modeling documents the processes and offers means for managing activities accordingly. BPMN 2.0 proved suitable for this purpose as it is intuitive to use, sufficiently rich in expressivity, flexible, and allows inclusion of diverse types of human and information system related tasks. However, being a relatively new method the tool support is, unfortunately, limited and insufficient especially considering process execution capabilities.

For engineering process modeling it is suggested to first define smaller processes that can then be used as building blocks in implementing processes on a larger scale. Internal processes within companies are typically stable and can be standardized to be exposed as engineering service offerings. Processes in engineering consortiums tend to be agile and flexible process management is required. Concerning engineering management, e.g. in virtual enterprises, the interaction processes as well as interfaces of information exchange are of primary importance. It is acknowledged that processes can include unexpected events that can be challenging to foresee and include in process definitions. It is an interesting question whether these unforeseen turns are inevitable or a result of poorly managed engineering.

The service based operation in combination with business process modeling can introduce means for managing and improving engineering processes. Information system supported business process execution automatically documents engineering performed, and collects metadata information that can be used to analyse and further improve the processes. Process execution also provides means for automation of engineering processes by reducing manual work and information exchange. In some cases simple tasks can even be fully automated.

Collaborative O&M processes could be improved if mutually valuable data could be shared and integrated. The information could give more insight into operations as well as improve capabilities of providing and consuming e.g. maintenance services. An information system architecture based on services can increase this information exchange. The information content, however, still needs consolidation and standardization efforts in order for service interfaces to be fully interoperable.

A proof of concept O&M service prototype has been discussed in which a SOA infrastructure is used to integrate information systems as Web services. The services are composed into service orchestrations according to the maintenance processes. For this a business process modeling approach has been adopted. BPMN is used to model maintenance processes that are then transformed to executable WS-BPEL.

The approach using business process models as the baseline for developing O&M applications represents a top-down development method. This advances development of applications driven by business and information exchange needs. BPMN based modeling of O&M processes and composition of services to executable processes has shown to be efficient in maintaining the required flexibility when reconfiguring process definitions. A service based operation model in O&M also promotes standardization of data, interfaces of tools and applications as well as the operating practices.

6 Summary of the Included Publications

The thesis includes seven publications. In this chapter the publications are summarized and the author's contribution to each of the publications is defined.

[P1]

The paper discusses current practices in process and control application design, and presents some information exchange challenges between related engineering disciplines. In the paper, attention is given to defining requirements for improving information transfer and utilization of a shared consolidating information model. The use of model-driven methods in control application development is conceptualized and the applicability and benefits for the industrial control domain are discussed. The initial experiences from the AUKOTON development approach are briefly mentioned as well.

The author is responsible for planning the paper and is the main author. Section 3 concerning Current Design Practices is a joint collaboration between the authors. The author is responsible for section 4, Defining Requirements for Enhanced Application Development, and section 5, Example Process and Experiences.

[P2]

The paper introduces and motivates the improvements made to the UML Automation Profile (UML AP). The AUKOTON model-driven development approach for control application development is presented. The approach is based on UML AP and the activities for requirements modeling, functional modeling, and platform dependent modeling are detailed. UML AP Tool support, including model transformations and modeling of execution platforms, is also presented. The development approach is demonstrated with an example evaluating and discussing the features of the approach.

The author is the main author of the paper and responsible for the paper in general including the related work and the discussion. The modeling concepts, the development approach, and the tools and plug-ins presented in the paper have been developed as a collaborative effort. For this the main contributions of the author are related to the modeling concepts, the model-driven development approach, and the information flow in the application development process. The second author of the paper is the main responsible for the parts concerning the tool support and for modeling and development related to platform specific features.

[P3]

The paper conceptualizes the use of OWL ontologies in combination with metamodel based models in model-driven development. The differences between ontology development and metamodeling are compared. An approach is then presented that aligns metamodels with a domain ontology and the model instances with an ontology containing the individuals. Transformations from UML AP metamodels and models to corresponding ontologies are considered. The results and experiences from developing a XSL based transformation are then discussed and the future use cases are outlined.

The author is the main author of the paper and responsible for planning the paper, implementing all software prototypes, and writing the paper.

[P4]

The paper proposes methods for applying OWL ontology based knowledge and reasoning as a supplementary means to support engineering. Engineering knowledge used during design is discussed and a coarse classification of storing this information in ontologies is proposed. The knowledge in ontologies is then used for reasoning on models. An UML AP based example is given to demonstrate model analysis and identification of structures that can be used to aid the designer in his work. A method is also presented for concurrently maintaining a semantic knowledge base of the models under design. Examples of its utilization in IDE integrations are given for integrating a model analysis service and linking supporting information. The applicability of the proposed methods is evaluated and requirements for using ontology knowledge in control application development are discussed.

The author is the main author of the paper and responsible planning the paper, implementing all software prototypes, and writing the paper.

[P5]

The paper discusses the concept of using a service based model for sharing engineering information and organizing engineering activities in control application development and related engineering disciplines. Current engineering practices and challenges are discussed from the point of view of accessing and utilizing engineering data required in performing the tasks. A shared Web service framework is proposed for improving information exchange and system integration. The engineering offerings are organized into service units exposed as programmatically invokable services. The purpose of the concept is then to be able to compose engineering services into workflows

corresponding to the engineering processes. The approach and the challenges are discussed for engineering but also for operation and maintenance purposes.

The author is the main author of the paper and responsible for planning and writing the paper.

[P6]

In the paper, a SOA service infrastructure is developed that as a service bus integrates and mediates both information sources and engineering services. Business process modeling is proposed as a management method to specify engineering processes. BPMN 2.0 is evaluated as a modeling notation with examples from typical engineering tasks. The automation of the business processes is implemented using process execution and utilization of the service bus infrastructure. The example processes demonstrate integration to shared plant information data, the use of service registries and mediated access to engineering services, and event based process execution.

The author is the main author of the paper and responsible for planning the paper, implementing all software prototypes, and writing the paper.

[P7]

The paper proposes an approach based on business process modeling for developing applications to support operation and maintenance. BPMN business process models are used to describe maintenance processes, and utilized as guidelines for implementing integrations and information flow in information systems. The benefits of SOA integrations are discussed from the point of view of composing services and integrating information sources into WS-BPEL orchestrations according to the business process definitions. The approach is argued to deliver required flexibility in service composition and reconfiguration of processes. The approach is demonstrated with a condition monitoring example utilizing existing engineering data in plant information systems and dynamic field device lookup for condition data. Another example is given in which based on user input environmental data available on the Web is gathered and processed to estimate the environmental footprint of devices.

The author is the main author of the paper and responsible for planning the paper. The author is responsible for the paper content with the exception of documenting and implementing the exemplary processes which was carried out by the second author. The author participated in the design of the exemplary processes and the prototypes.

7 Conclusions

The thesis considers control application development from the point of view of information model content, methods enhancing development and engineering support, as well as services and processes to improve information exchange and engineering process management. This chapter summarizes the thesis, the research questions, the limitations, and finally outlines future work.

7.1 Thesis Summary

Requirements for information content in control application models are discussed with respect to current practices and the needs and requirements of the domain. Based on this the UML AP modeling method is proposed for modeling control applications using domain specific concepts. UML AP includes modeling constructs for requirements modeling that enable concerns of various stakeholders and related disciplines to be captured in the requirements model. For modeling control functionality UML AP provides constructs on a suitable level of abstraction for the task at hand. The constructs have been designed with domain concepts in mind, and the functional modeling is of a platform independent nature. To support the use of existing well-proven control system platforms means for modeling these platforms are presented.

In order to enhance the use of information in control application models OWL ontologies are applied. For this an approach is presented that aligns the metamodel, i.e. the modeling concepts, and application instance models with corresponding domain ontologies and instance ontologies, respectively. Two separate transformation approaches are then presented. The first one is aimed for platform independent model information management, and the second one for concurrent and application development integrated use, e.g. in an IDE.

Model-driven methods promote the use of models as primary objects of development and the methods that are considered for control application engineering. From a control applications modeling perspective requirements are specified for implementing development methods and information flow in related engineering processes. A development approach is then presented that - using UML AP Requirement constructs - enables related engineering to be considered at first hand. The approach includes tool support for automating the consolidation of source data to be used in the development process. Moving to functional modeling the approach does not require development to become biased of implementation platforms until the platform is decided. This promotes

designing and using reusable, platform independent solutions. Automatic transformations are also included that assist in creating functional models from requirements. Similar transformations bind the functional models to implementation platforms. The evaluation presented and the results in the included publications suggest that model-driven development based on UML AP can improve development of control applications.

The use of Semantic Web ontologies is proposed as a means to support engineering tasks. The use of engineering knowledge stored in ontologies is discussed and a categorization is presented. Using general engineering knowledge in OWL ontologies, and an ontology mapping the concepts to those of UML AP, elements and structures of interest are identified in functional models. The semantic descriptions are used for reasoning on models e.g. to detect flaws or error-prone designs. For this an analysis service is developed that successfully demonstrated the use of ontology semantics in model classification and analysis. The semantic inferences and generalizations of models under design are also used to provide concurrent engineering support in the form of automatically linking related material.

Improvements to the engineering processes can be considered when the model information content is structured and specified, and the development methods are defined. The use of services is proposed as a means for organizing engineering tasks and offerings of e.g. specialist services. Requirements for using these kinds of services are discussed and the concept of a consolidating service framework is presented. Service-oriented architecture and Web services are proposed as a means for system integration. A proof of concept service infrastructure is developed to demonstrate this. The infrastructure is based on a service bus acting as a mediator and an integrator of information sources such as model repositories and plant information models, engineering services, adapter services, a service registry, and process management tools.

BPMN is proposed for defining the engineering processes, and to connect engineering tasks organized and exposed as services. The process definitions are used to compose services into engineering workflows and to manage the engineering activities. The developed service infrastructure is used for this purpose as the integration platform. Business process execution is applied to improve gathering results and to automate some of the tasks in the engineering processes. Examples are given for use of services in which both manual work and automatic service execution is performed.

The information models developed during engineering are of importance also during the operation of the plant, e.g. in maintenance. The use of this information is discussed from

the point of view of accessing it for later engineering and maintenance purposes. An application development approach supporting O&M is proposed based on business process modeling methods. In this case the business processes serve as definitions for developing supporting information systems corresponding to the actual processes. The aim is to provide flexible means for adaptation to changing circumstances of varying service providers and changing business environments. To facilitate integration of information systems SOA is considered as the basis for composing services into orchestrations using WS-BPEL. An example related to condition monitoring is presented to demonstrate the approach.

7.2 Research Questions Revisited

The linking of business strategy goals with software development efforts has been studied, and goal levels for business, software development and individual development projects have been defined by [5]. In their model a goal is realized by a strategy that is influenced and dependent on the context. The elements are hierarchical such that individual development project goals and strategies are sub goals and sub strategies of supporting software development level goals and strategies, and ultimately contribute to business level goals and strategies. In a similar manner the goals of this thesis build up strategies in order to eventually contribute to improving the way businesses operate in engineering industrial control systems. The research performed in each topic corresponding to a research question is built on and utilized in the following research question.

The research questions have been covered and discussed in chapters 3, 4, and 5, respectively. Following is a summary of the answers to the research questions.

RQ1: What kind of engineering information is required in development for the industrial control application domain? What requirements are there on concepts and constructs used in modeling from a distributed development perspective? How can interoperability of models and concepts be improved to support development during the engineering lifecycle?

Engineering of control applications depends on process engineering, piping and instrumentation, and increasingly also on intelligent equipment such as field devices and other applications that are integrated e.g. as subsystems. To utilize this information effectively it needs to be structured and preferably be based on standards and agreed practices. This enhances development of automatic import functions and reduces the risk of interoperability concerns and disambiguation of information.

The control application modeling constructs need to be able to integrate this source information and UML AP is proposed to cover these aspects. UML AP provides orthogonal constructs for modeling requirements, functions, and platform specific features. Concerning transfer of model information UML AP endorses organizing models according to development phases for modeling of requirements, functionality, and execution platform features. Structuring of models into packages with defined interaction points enhances communicating designs in distributed engineering.

The use of standard based methods for which tool independent access can be developed promotes information exchange in development. To further improve interoperability and designs semantics methods for using OWL ontologies as a supplement is proposed. The semantic descriptions are used as an additional layer on top of the UML AP models.

RQ2: What are the characteristics of development methods that enhance distributed engineering in the domain? Can development be systematized with model-driven methods to improve transfer of multi-disciplinary information and what are its implications on application development? How can engineering concepts of the systematized process be used to support development?

To enhance distributed engineering, including control application development, the development methods need to be based on information models that are capable of expressing the required information explicitly, and are interpretable and unambiguous to the designers. In order to utilize these models the development approach needs to clearly define the responsibilities and requirements for each phase of development. This is required in order to be able to distribute engineering work, provide models with sufficient data and manage the development process. To improve efficiency of engineering both automated data processing and model transformations are required.

The AUKOTON development approach fulfils many of these requirements. The development approach endorses platform independent functional modeling which can be seen as a factor in improving understandability and reuse of modelled solutions. However, sufficient means have not been developed to fully support collaboration and integration of designs of distributed engineering efforts concerning models that focus on the same development phase and same set of objects. The use of the AUKOTON development approach would also put pressure on related engineering disciplines to systematize engineering efforts and use standardized information models in communicating source data.

To improve interoperability and understandability of concepts used in engineering Semantic Web technologies can be used to provide additional semantics to structures

and deliver concurrent engineering support. Engineering knowledge can be used in semantic analysis of models to increase quality e.g. to assist the developer to focus on critical tasks.

RQ3: What are the requirements for automating engineering processes? How can engineering processes be managed and automated using services and business process modeling methods? Are the concepts and development methods proposed above applicable in improving management of distributed engineering activities and automating some of the tasks?

Engineering processes of control application development are a viable candidate for using services to promote design information exchange in engineering processes, and be managed and automated using information system implementation. This is under the precondition that information models support sufficient and unambiguous information exchange (RQ1), and development methods allow distribution of work processes with defined interaction points (RQ2). Then engineering offerings can be organized into services.

A service infrastructure can be used to mediate information system access and the use of services representing engineering offerings. The engineering workflows can be defined using means of business process modeling; this was demonstrated using BPMN. The execution of these process definitions facilitates automation of tasks for information gathering but also enables simple tasks to be performed fully automatically. The process definitions also assist in managing engineering activities and by collecting process execution metadata the processes can be improved in the long run.

Examples have been presented where UML AP control application models are used in engineering processes. The use of services for development services e.g. the AUKOTON development approach has also been described. The Semantic Web extensions were successfully utilized in semantic analysis of UML AP models with the help of transformation services provided by the service bus infrastructure.

7.3 Limitations and Future Work

The methods and approaches proposed in this thesis have been implemented and experimentally verified on a prototype level. The research questions are multidimensional and the results obtained by constructive research consider improvements mainly from a qualitative point of view based on evaluation criteria. It

can be argued that the results give a good indication of the suitability of the developed methods and the potential improvement in real production environments.

In order to get quantitative measures of improved efficiency, quantitative quality measures of developed applications, increased solution reuse, and ease of development, the methods and approaches would have to be implemented in commercial engineering environments and applied to large projects. It would also require a number of projects before the real engineering environments have evolved to fully enable using development approaches and operation principles discussed in this thesis.

Concerning future work for model information content feasible standardization of engineering data is required to further advance information exchange. This includes both semantics of concepts as well as fine-grained computer processable information models. The interoperability of information could be improved with e.g. adoption of Linked Data principles. However, as the Semantic Web descriptions do not include any means to ensure integrity of information some other means e.g. for validation would be useful. The use of models within applications being executed was not considered in this thesis but it is a viable direction for future research.

For methods used in development processes, research is required related to reuse of previously engineered solutions, and broadening the engineering scope to better take into account other points of view such as information networks, system integrations, security and safety, and upper level system functionality. The benefits of the semantics aware engineering IDE are more evident when larger structures and compositions are considered and a plausible use case is applying it to the selection and use of design patterns. The IDE intelligence could also be used to store the context of engineering actions in relation to more advanced work support features, e.g. related to searches or annotating designs.

Considering processes and services it is challenging, if not impossible, to specify models and development methods that at all times explicitly and unambiguously express necessary information required in engineering. Therefore development methods would need further research on how to cope with e.g. missing information in engineering processes. Intelligence to the execution of business processes could assist in dynamic process reconfigurations. However, the engineering service definitions would need specification of how work tasks are to be defined. The service bus would also benefit from standardized information, and methods offering more intelligent means to data integration and concept alignment could be developed as well.

Bibliography

[1] Albert, M., Cabot, J., Gómez, C. and Pelechano, V. Generating Operation Specifications from UML Class Diagrams: A Model Transformation Approach. *Data & Knowledge Engineering*, 2011, vol. 70, no. 4, pp. 365-389.

[2] Baines, T.S., et al. State-of-the-Art in Product-Service Systems. *Proceedings of the Institution of Mechanical Engineers -- Part B -- Engineering Manufacture*, 2007, vol. 221, no. 10, pp. 1543-1552.

[3] Bangemann, T., Rebeuf, X., Reboul, D., Schulze, A., Szymanski, J., Thomesse, J., Thron, M. and Zerhouni, N. PROTEUS — Creating Distributed Maintenance Systems through an Integration Platform. *Computers in Industry*, 2006, vol. 57, no. 6, pp. 539-551.

[4] Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S. and Sriram, R.D. OntoSTEP: Enriching Product Model Data using Ontologies. *Computer-Aided Design*, 2012, vol. 44, no. 6, pp. 575-590.

[5] Basili, V.R., Lindvall, M., Regardie, M., Seaman, C., Heidrich, J., Munch, J., Rombach, D. and Trendowicz, A. Linking Software Development and Business Strategy through Measurement. *Computer*, 2010, vol. 43, no. 4, pp. 57-65.

[6] Batres, R., West, M., Leal, D., Price, D., Masaki, K., Shimada, Y., Fuchino, T. and Naka, Y. An Upper Ontology Based on ISO 15926. *Computers & Chemical Engineering*, 2007, vol. 31, no. 5–6, pp. 519-534.

[7] Bayer, B. and Marquardt, W. Towards Integrated Information Models for Data and Documents. *Computers & Chemical Engineering*, 2004, vol. 28, no. 8, pp. 1249-1266.

[8] Bergman, R., Borden, C.S. and Zendejas, S. Automated Workflow for Engineering Services. In: IEEE Aerospace Conference Proceedings, 2002, vol. 5, pp. 2553-2567.

[9] Berners-Lee, T., Hendler, J. and Lassila, O. The Semantic Web. *Scientific American*, 2001, vol. 284, no. 5, pp. 34-43.

[10] Biffl, S., Schatten, A. and Zoitl, A. Integration of Heterogeneous Engineering Environments for the Automation Systems Lifecycle. In: 7th IEEE International Conference on Industrial Informatics, INDIN 2009, pp. 576-581.

[11] Bizer, C., Heath, T. and Berners-Lee, T. Linked Data - the Story so Far. *International Journal on Semantic Web and Information Systems*, 2009, vol. 5, no. 3, pp. 1-22.

[12] Bollati, V.A., Vara, J.M., Jiménez, Á. and Marcos, E. Applying MDE to the (Semi-)Automatic Development of Model Transformations. *Information and Software Technology*, 2013, vol. 55, no. 4, pp. 699-718.

[13] Bonfe, M. and Fantuzzi, C. Design and Verification of Industrial Logic Controllers with UML and Statecharts. In: Proceedings of the 2003 IEEE Conference on Control Applications, CCA 2003, June 2003, vol. 2, pp. 1029-1034.

[14] Bracht, U. and Masurat, T. The Digital Factory between Vision and Reality. *Computers in Industry*, 2005, vol. 56, no. 4, pp. 325-333.

[15] Breslin, J.G., O'Sullivan, D., Passant, A. and Vasiliu, L. Semantic Web Computing in Industry. *Computers in Industry*, 2010, vol. 61, no. 8, pp. 729-741.

[16] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J. and Brandic, I. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, 2009, vol. 25, no. 6, pp. 599-616.

[17] Carroll, J.J., Bizer, C., Hayes, P. and Stickler, P. Named Graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2005, vol. 3, no. 4, pp. 247-267.

[18] Chinosi, M. and Trombetta, A. BPMN: An Introduction to the Standard. *Computer Standards & Interfaces*, 2012, vol. 34, no. 1, pp. 124-134.

[19] Chiron, F. and Kouiss, K. Design of IEC 61131-3 Function Blocks using SysML. In: 2007 Mediterranean Conference on Control & Automation, MED '07, June 27-29 2007, pp. 1-5.

[20] Ciccozzi, F., Cicchetti, A. and Sjödin, M. Round-Trip Support for Extra-Functional Property Management in Model-Driven Engineering of Embedded Systems. *Information and Software Technology*. 2012. DOI: 10.1016/j.infsof.2012.07.014.

[21] Crnkovic, G. *Constructive Research and Info-Computational Knowledge Generation.* Model-Based Reasoning in Science and Technology. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2010, vol. 314, pp. 359-380.

[22] Domínguez-Mayo, F.J., Escalona, M.J., Mejías, M., Ross, M. and Staples, G. Quality Evaluation for Model-Driven Web Engineering Methodologies. *Information and Software Technology*, 2012, vol. 54, no. 11, pp. 1265-1282.

[23] Drath, R., Luder, A., Peschke, J. and Hundt, L. AutomationML - the Glue for Seamless Automation Engineering. In: 13th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2008, September 2008, pp. 616-623.

[24] Dubinin, V., Vyatkin, V. and Pfeiffer, T. Engineering of Validatable Automation Systems Based on an Extension of UML Combined with Function Blocks of IEC 61499. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 2005, pp. 3996-4001.

[25] Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005 ISBN 0131858580.

[26] Farail, P., Gaufillet, P., Canals, A., Le Camus, C., Sciamma, D., Michel, P., Crégut, X. and Pantel, M. The TOPCASED Project: A Toolkit in OPen Source for Critical Aeronautic SystEms Design. In: Embedded Real Time Software, ERTS 2006.

[27] Fielding, R.T. *REST: Architectural Styles and the Design of Network-Based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[28] France, R. and Rumpe, B. Model-Driven Development of Complex Software: A Research Roadmap. In: Future of Software Engineering, 2007, February 2007, pp. 37-54.

[29] Grabmair, G., Zoitl, A., Strasser, T. and Froschauer, R. Modelling Real-Time Constraints regarding Reconfiguration Aspects for IEC 61499 Control Applications. In: 5th IEEE International Conference on Industrial Informatics, INDIN'07, June 2007, vol. 2, pp. 1089-1094.

[30] Graube, M., Pfeffer, J., Ziegler, J. and Urbas, L. Linked Data as Integrating Technology for Industrial Data. In: 14th International Conference on Network-Based Information Systems, NBiS 2011, pp. 162-167.

[31] Happel, H. and Seedorf, S. Applications of Ontologies in Software Engineering. In: 2nd International Workshop on Semantic Web Enabled Software Engineering, SWESE'06, November 2006.

[32] Hästbacka, D., Laitinen, O., Tommila, T. and Kuikka, S. Implementing a Work Support and Training Tool for Control Engineers. In: 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2007, September 6-8, 2007, pp. 512-517.

[33] Hästbacka, D. and Kuikka, S. *Semantics and Reasoning for Control Application Engineering Models.* Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7267, pp. 647-655.

[34] Hausenblas, M. Exploiting Linked Data to Build Web Applications. *Internet Computing, IEEE*, 2009, vol. 13, no. 4, pp. 68-73.

[35] Hausladen, I. and Bechheim, C. E-Maintenance Platform as a Basis for Business Process Integration. In: 2nd IEEE International Conference on Industrial Informatics, INDIN 2004, pp. 46-51.

[36] Hettel, T., Lawley, M. and Raymond, K. Model Synchronisation: Definitions for Round-Trip Engineering. In: Proceedings of the 1st International Conference on Theory and Practice of Model Transformations, 2008, pp. 31-45.

[37] Hong-Seok Na, O-Hoon Choi and Jung-Eun Lim. A Method for Building Domain Ontologies Based on the Transformation of UML Models. In: 4th International Conference on Software Engineering Research, Management and Applications, 2006, pp. 332-338.

[38] Hou, Z. and Wang, Z. From Model-Based Control to Data-Driven Control: Survey, Classification and Perspective. *Information Sciences*, 2013, vol. 235, pp. 3-35.

[39] Hsieh, S., Lin, H., Chi, N., Chou, K. and Lin, K. Enabling the Development of Base Domain Ontology through Extraction of Knowledge from Engineering Domain Handbooks. *Advanced Engineering Informatics*, 2011, vol. 25, no. 2, pp. 288-296.

[40] Hussain, T. and Frey, G. Defining IEC 61499 Compliance Profiles using UML and OCL. In: 5th IEEE International Conference on Industrial Informatics, INDIN 2007, 23-27 June 2007, vol. 2, pp. 1157-1162.

[41] Hutchinson, J., Kotonya, G., Walkerdine, J., Sawyer, P., Dobson, G. and Onditi, V. The Challenge of Evolving Existing Systems to Service-Oriented Architectures. In: 5th IEEE International Conference on Industrial Informatics, INDIN 2007, 23-27 June 2007, vol. 2, pp. 773-778.

[42] IEC. *IEC 62424: Representation of Process Control Engineering - Requests in P&I Diagrams and Data Exchange between P&ID Tools and PCE-CAE Tools.* International Electrotechnical Commission, 2008.

[43] IEC. *IEC 61499, Function Blocks, Part 1-Part 4.* International Electrotechnical Commission, 2005.

[44] IEC. *IEC 61131-3: Programmable Controllers - Part 3: Programming Languages.* International Electrotechnical Commission, 2003.

[45] Isermann, R. Perspectives of Automatic Control. *Control Engineering Practice*, 2011, vol. 19, no. 12, pp. 1399-1407.

[46] ISO. *ISO 15926-1 Industrial Automation Systems and Integration — Integration of Life-Cycle Data for Process Plants Including Oil and Gas Production Facilities — Part 1: Overview and Fundamental Principles.* International Organization for Standardization, 2004.

[47] Iung, B., Levrat, E., Marquez, A.C. and Erbe, H. Conceptual Framework for e-Maintenance: Illustration by e-Maintenance Technologies and Platforms. *Annual Reviews in Control*, 2009, vol. 33, no. 2, pp. 220-229.

[48] Jouault, F., Allilaire, F., Bézivin, J. and Kurtev, I. ATL: A Model Transformation Tool. *Science of Computer Programming*, 2008, vol. 72, no. 1–2, pp. 31-39.

[49] Karaila, M. Evolution of a Domain Specific Language and its Engineering Environment - Lehman's Laws Revisited. In: Proceedings of the 9th OOPSLA Workshop on Domain-Specifc Modeling (DSM'09), 25-26 October 2009.

[50] Karnouskos, S. and Colombo, A.W. Architecting the Next Generation of Service-Based SCADA/DCS System of Systems. In: IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, pp. 359-364.

[51] Kim, B.C., Teijgeler, H., Mun, D. and Han, S. Integration of Distributed Plant Lifecycle Data using ISO 15926 and Web Services. *Annals of Nuclear Energy*, 2011, vol. 38, no. 11, pp. 2309-2318.

[52] Knorr, M., Alferes, J.J. and Hitzler, P. Local Closed World Reasoning with Description Logics Under the Well-Founded Semantics. *Artificial Intelligence*, 2011, vol. 175, no. 9–10, pp. 1528-1554.

[53] Knublauch, H., Hendler, J.A. and Idehen, K. *SPIN - Overview and Motivation.* W3C Member Submission. 2011.

[54] Lartigau, J., Lanshun Nie, Xiaofei Xu, Dechen Zhan and Tehani Mou. Scheduling Methodology for Production Services in Cloud Manufacturing. In: International Joint Conference on Service Sciences, IJCSS 2012, pp. 34-39.

[55] Levery, M. Outsourcing Maintenance-a Question of Strategy. *Engineering Management Journal*, 1998, vol. 8, no. 1, pp. 34-40.

[56] March, S.T. and Smith, G.F. Design and Natural Science Research on Information Technology. *Decis.Support Syst.*, 1995, vol. 15, no. 4, pp. 251-266.

[57] Marcos, M., Estévez, E., Gangoiti, U., Sarachaga, I. and Barandiarán, J. UML Modelling of Industrial Distributed Control Systems. In: Proceedings of the 6th Portuguese Conference on Automatic Control, Controlo 2004, June 2004.

[58] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J. and Ghalsasi, A. Cloud Computing — the Business Perspective. *Decision Support Systems*, 2011, vol. 51, no. 1, pp. 176-189.

[59] Martini, A. *Merge of Models: An XMI Approach.* Master's Thesis, Department of Computer Science, Faculty of Engineering LTH, Lund University, 2010.

[60] Mazanek, S. and Hanus, M. Constructing a Bidirectional Transformation between BPMN and BPEL with a Functional Logic Programming Language. *Journal of Visual Languages & Computing*, 2011, vol. 22, no. 1, pp. 66-89.

[61] Miorandi, D., Sicari, S., De Pellegrini, F. and Chlamtac, I. Internet of Things: Vision, Applications and Research Challenges. *Ad Hoc Networks*, 2012, vol. 10, no. 7, pp. 1497-1516.

[62] Mokos, K., Meditskos, G., Katsaros, P., Bassiliades, N. and Vasiliades, V. Ontology-Based Model Driven Engineering for Safety Verification. In: 36th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2010, pp. 47-54.

[63] Mordinyi, R., Moser, T., Winkler, D. and Biffl, S. Navigating between Tools in Heterogeneous Automation Systems Engineering Landscapes. In: IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp. 6178-6184.

[64] Moser, T. and Biffl, S. Semantic Integration of Software and Systems Engineering Environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2012, vol. 42, no. 1, pp. 38-50.

[65] OASIS. *Devices Profile for Web Services Version 1.1*. D. Driscoll, and A. Mensch eds., Organization for the Advancement of Structured Information Standards, 2009.

[66] OASIS. *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*. Organization for the Advancement of Structured Information Standards, 2007.

[67] OASIS. *Reference Model for Service Oriented Architecture 1.0*. Organization for the Advancement of Structured Information Standards, 2006.

[68] OMG. *Business Process Model and Notation (BPMN) Version 2.0*. Object Management Group, 2011.

[69] OMG. *Object Constraint Language, v 2.2*. Object Management Group, 2010.

[70] OMG. *Business Process Modeling Notation (BPMN) Version 1.2*. Object Management Group, 2009.

[71] OMG. *Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT), Version 1.0*. Object Management Group, 2008.

[72] OMG. *System Modeling Language (SysML), Version 1.0*. Object Management Group, 2007.

[73] OMG. *XML Metadata Interchange (XMI), Version 2.1.1*. Object Management Group, 2007.

[74] OMG. *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, Version 1.0*. Object Management Group, 2006.

[75] OMG. *Unified Modeling Language Specification 2.0: Infrastructure*. Object Management Group, 2006.

[76] OMG. *UML Profile for Schedulability, Performance, and Time Specification*. Object Management Group, 2005.

[77] OMG. *Unified Modeling Language Specification 2.0: Superstructure*. Object Management Group, 2005.

[78] OMG. *Meta Object Facility (MOF) 2.0 Core Final Adopted Specification*. Object Management Group, 2004.

[79] OMG. *Model Driven Architecture (MDA) Guide*. Object Management Group, 2003.

[80] Oren, E., Heitmann, B. and Decker, S. ActiveRDF: Embedding Semantic Web Data into Object-Oriented Languages. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2008, vol. 6, no. 3, pp. 191-202.

[81] Ottensooser, A., Fekete, A., Reijers, H.A., Mendling, J. and Menictas, C. Making Sense of Business Process Descriptions: An Experimental Comparison of Graphical and Textual Notations. *Journal of Systems and Software*, 2012, vol. 85, no. 3, pp. 596-606.

[82] Owen, C. Design Research: Building the Knowledge Base. *Journal of the Japanese Society for the Science of Design*, 1997, vol. 5, no. 2, pp. 36-45.

[83] Panjaitan, S.D. and Frey, G. Development Process for Distributed Automation Systems Combining UML and IEC 61499. *Int.J.Manufacturing Research*, 2007, vol. 2, no. 1, pp. 1-20.

[84] Parreiras, F.S. and Staab, S. Using Ontologies with UML Class-Based Modeling: The TwoUse Approach. *Data & Knowledge Engineering*, 2010, vol. 69, no. 11, pp. 1194-1207.

[85] Patel-Schneider, P.F. and Horrocks, I. A Comparison of Two Modelling Paradigms in the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2007, vol. 5, no. 4, pp. 240-250.

[86] Persson, A., Gustavsson, H., Lings, B., Lundell, B., Mattsson, A. and Ärlig, U. OSS Tools in a Heterogeneous Environment for Embedded Systems Modelling: An Analysis of Adoptions of XMI. In: Proceedings of the 5th Workshop on Open source software engineering, 2005, pp. 1-4.

[87] Pilone, D. and Pitman, N. *UML 2.0 in a Nutshell*. First ed. O'Reilly Media, 2005 ISBN 81-8404-002-4.

[88] Pratt, M.J. Introduction to ISO 10303 — the STEP Standard for Product Data Exchange. *J.Comput.Info.Sci.Eng.*, 2001, vol. 1, no. 1, pp. 102-103.

[89] Purao, S. *Design Research in the Technology of Information Systems: Truth Or Dare.* [Working Paper].GSU Department of CIS, Atlanta, 2002.

[90] Puttonen, J., Lobov, A., Soto, M.A.C. and Lastra, J.L.M. A Semantic Web Services-Based Approach for Production Systems Control. *Advanced Engineering Informatics*, 2010, vol. 24, no. 3, pp. 285-299.

[91] Ramos-Hernandez, D.N., Fleming, P.J. and Bass, J.M. A Novel Object-Oriented Environment for Distributed Process Control Systems. *Control Engineering Practice*, 2005, vol. 13, no. 2, pp. 213.

[92] Rauhamäki, J., Laitinen, O., Sierla, S. and Kuikka, S. The Role of User Guidance in the Industrial Adoption of MDE Approach. *Electronic Communications of the EASST*, 2010, vol. 34.

[93] Rauschecker, U. and Stohr, M. Using Manufacturing Service Descriptions for Flexible Integration of Production Facilities to Manufacturing Clouds. In: 18th International ICE Conference on Engineering, Technology and Innovation, ICE 2012, pp. 1-10.

[94] Ritala, T. and Kuikka, S. UML Automation Profile: Enhancing the Efficiency of Software Development in the Automation Industry. In: 5th IEEE International Conference on Industrial Informatics, INDIN 2007, 23-27 June, 2007, vol. 2, pp. 885-890.

[95] Robles, K., Fraga, A., Morato, J. and Llorens, J. Towards an Ontology-Based Retrieval of UML Class Diagrams. *Inf.Softw.Technol.*, 2012, vol. 54, no. 1, pp. 72-86.

[96] Savioja, P., Salo, L., Laitinen, O., Hästbacka, D., Juden, T. and Valve, V. *Defining a Work Support and Training Tool for Automation Design Engineers.* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, 2007, vol. 4562, pp. 174-183.

[97] Schamai, W. *Modelica Modeling Language (ModelicaML): A UML Profile for Modelica.* Linköping University, Department of Computer and Information Science, Linköping University Electronic Press. 2009.

[98] Schneider, R. and Marquardt, W. Information Technology Support in the Chemical Process Design Life Cycle. *Chemical Engineering Science*, 2002, vol. 57, no. 10, pp. 1763-1792.

[99] Seuranen, T., Karhela, T. and Hurme, M. Automated Process Design using Web-Service Based Parameterised Constructors. In: 38th European Symposium of the Working Party on Computer Aided Process Engineering, Computer Aided Chemical Engineering, 2005, vol. 20, pp. 1639-1645.

[100] Shourong Lu, Halang, W.A. and Lichen Zhang. A Component-Based UML Profile to Model Embedded Real-Time Systems Designed by the MDA Approach. In: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, August 2005, pp. 563-566.

[101] Simensen, J., Johnsson, C. and Årzén, K.E. A Multiple-View Batch Plant Information Model. *Computers & Chemical Engineering*, 1997, vol. 21, Supplement, pp. S1209-S1214.

[102] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. and Katz, Y. Pellet: A Practical OWL-DL Reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2007, vol. 5, no. 2, pp. 51-53.

[103] Soylu, A. and De Causmaecker, P. Merging Model Driven and Ontology Driven System Development Approaches Pervasive Computing Perspective. In: 24th International Symposium on Computer and Information Sciences, ISCIS 2009, September 14-16, 2009, pp. 730-735.

[104] Spohrer, J., Vargo, S.L., Caswell, N. and Maglio, P.P. The Service System is the Basic Abstraction of Service Science. In: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, 2008, pp. 104-104.

[105] Spring, M. and Araujo, L. Beyond the Service Factory: Service Innovation in Manufacturing Supply Networks. *Industrial Marketing Management* DOI: 10.1016/j.indmarman.2012.11.006.

[106] Staab, S., Walter, T., Gröner, G. and Parreiras, F.S. *Model Driven Engineering with Ontology Technologies.* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, 2010, vol. 6325, pp. 62-98.

[107] Stahl, T., Voelter, M. and Czarnecki, K. *Model-Driven Software Development: Technology, Engineering, Management.* John Wiley & Sons, 2006 ISBN 0470025700.

[108] Strasser, T., Rooker, M., Hegny, I., Wenger, M., Zoitl, A., Ferrarini, L., Dede, A. and Colla, M. A Research Roadmap for Model-Driven Design of Embedded Systems for Automation Components. In: 7th IEEE International Conference on Industrial Informatics, INDIN 2009, June 2009, pp. 564-569.

[109] Strasser, T., Sünder, C. and Valentini, A. Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector. In: 6th IEEE International Conference on Industrial Informatics, INDIN 2008, July 2008, pp. 1120-1125.

[110] Sünder, C., Zoitl, A., Favre-Bulle, B., Strasser, T., Steininger, H. and Thomas, S. *Towards Reconfiguration Applications as Basis for Control System Evolution in Zero-Downtime Automation Systems.* Intelligent Production Machines and Systems. Elsevier Science Ltd, 2006, pp. 523-528.

[111] Thramboulidis, K., Perdikis, D. and Kantas, S. Model Driven Development of Distributed Control Applications. *The International Journal of Advanced Manufacturing Technology*, 2007, vol. 33, no. 3, pp. 233-242.

[112] Thramboulidis, K.C., Doukas, G. and Koumoutsos, G. A SOA-Based Embedded Systems Development Environment for Industrial Automation. *EURASIP J.Embedded Syst.*, 2008, vol. 2008, pp. 3:1-3:15.

[113] Tianfield, H. and Unland, R. Towards Autonomic Computing Systems. *Engineering Applications of Artificial Intelligence*, 2004, vol. 17, no. 7, pp. 689-699.

[114] van Amerongen, J. and Breedveld, P. Modelling of Physical Systems for the Design and Control of Mechatronic Systems. *Annual Reviews in Control*, 2003, vol. 27, no. 1, pp. 87-117.

[115] Vargo, S.L. and Lusch, R.F. Evolving to a New Dominant Logic for Marketing. *Journal of Marketing*, 2004, vol. 68, no. 1, pp. 1-17.

[116] Vepsäläinen, T., Hästbacka, D. and Kuikka, S. Simulation Assisted Model-Based Control Development - Unifying UML AP and Modelica ML. In: 11th International Middle Eastern Simulation Multi Conference, MESM 2010, December 1-3, 2010, pp. 43-50.

[117] Vepsäläinen, T., Hästbacka, D. and Kuikka, S. A Model-Driven Tool Environment for Automation and Control Application Development - Transformation Assisted, Extendable Approach. In: Proceedings of 11th Symposium on Programming Languages and Software Tools and 7th Nordic Workshop on Model Driven Software Engineering, August 26-28 2009, pp. 315-329.

[118] Vepsäläinen, T., Hästbacka, D. and Kuikka, S. Tool Support for the UML Automation Profile - for Domain-Specific Software Development in Manufacturing. In: Proceedings of the 3rd International Conference on Software Engineering Advances, ICSEA 2008, pp. 43-50.

[119] Vepsäläinen, T., Sierla, S., Peltola, J. and Kuikka, S. Assessing the Industrial Applicability and Adoption Potential of the AUKOTON Model Driven Control Application Engineering Approach. In: 8th IEEE International Conference on Industrial Informatics, INDIN 2010, pp. 883-889.

[120] Vepsäläinen, T. and Kuikka, S. *Simulation-Based Development of Safety Related Interlocks.* Simulation and Modeling Methodologies, Technologies and Applications. Advances in Intelligent Systems and Computing. Springer Berlin Heidelberg, 2013, vol. 197, pp. 165-182.

[121] Vyatkin, V., Hanisch, H.-., Cheng Pang and Chia-Han Yang. Closed-Loop Modeling in Future Automation System Engineering and Validation. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 2009, vol. 39, no. 1, pp. 17-28.

[122] W3C. *OWL 2 Web Ontology Language Document Overview.* World Wide Web Consortium, 2009.

[123] W3C. *SOAP Version 1.2 Part 1: Messaging Framework.* World Wide Web Consortium, 2007.

[124] W3C. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.* World Wide Web Consortium, 2007.

[125] Walter, T., Parreiras, F. and Staab, S. An Ontology-Based Framework for Domain-Specific Modeling. *Software & Systems Modeling*, 2012, pp. 1-26.

[126] Walter, T., Silva Parreiras, F. and Staab, S. OntoDSL: An Ontology-Based Framework for Domain-Specific Languages. In: Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems, MODELS 2009, October 4-9, 2009, vol. 5795, pp. 408-422.

[127] Wicks, M.N. and Dewar, R.G. A New Research Agenda for Tool Integration. *Journal of Systems and Software*, 2007, vol. 80, no. 9, pp. 1569-1585.

[128] Wiesner, A., Morbach, J. and Marquardt, W. Information Integration in Chemical Process Engineering Based on Semantic Technologies. *Computers & Chemical Engineering*, 2011, vol. 35, no. 4, pp. 692-708.

[129] Witsch, D. and Vogel-Heuser, B. Close Integration between UML and IEC 61131-3: New Possibilities through Object-Oriented Extensions. In: 14th IEEE Conference on Emerging Technologies Factory Automation, ETFA 2009, September 22-26 2009, pp. 1-6.

[130] Xu, X. From Cloud Computing to Cloud Manufacturing. *Robotics and Computer-Integrated Manufacturing*, 2012, vol. 28, no. 1, pp. 75-86.

[131] Ying Cheng, Yue Zhang, Lin Lv, Jiarui Liu, Fei Tao and Lin Zhang. Analysis of Cloud Service Transaction in Cloud Manufacturing. In: 10th IEEE International Conference on Industrial Informatics, INDIN 2012, pp. 320-325.

[132] Zhang, L., Guo, H., Tao, F., Luo, Y.L. and Si, N. Flexible Management of Resource Service Composition in Cloud Manufacturing. In: 17th IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2010, pp. 2278-2282.

Publications

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-3098-2
ISSN 1459-2045